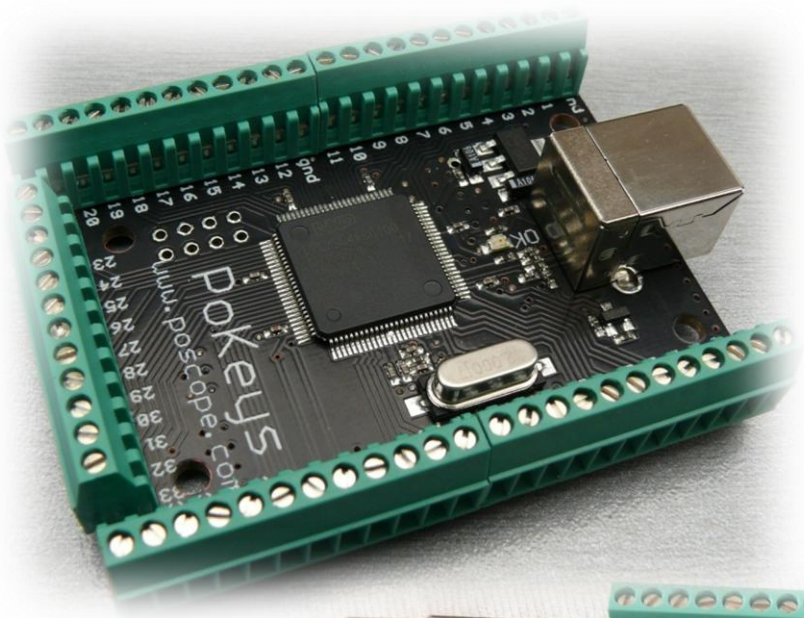
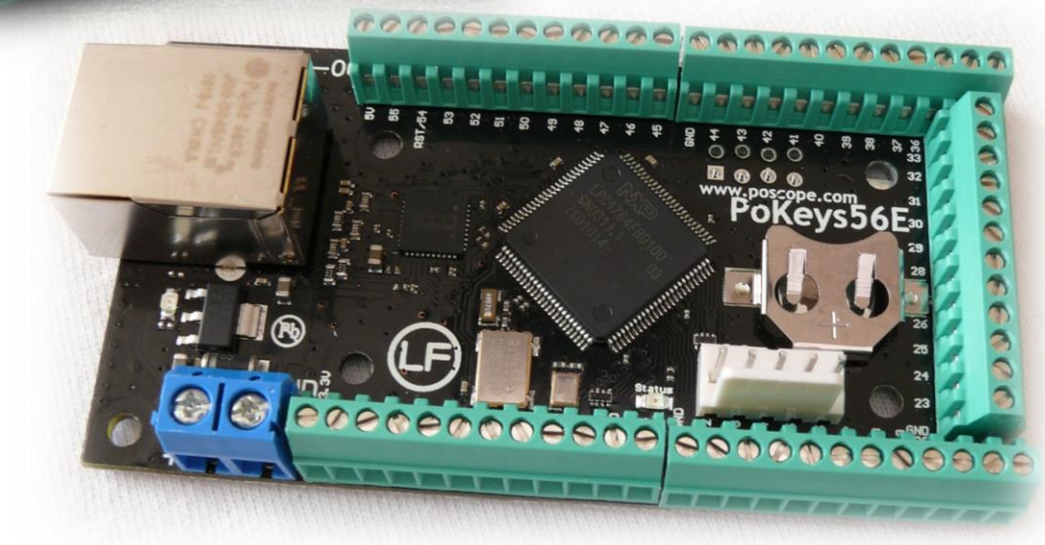
User's manual

Version: 1/10/2012





Please read the following notes

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice.
2. PoLabs does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of PoLabs products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of PoLabs or others. PoLabs claims the copyright of, and retains the rights to, all material (software, documents, etc.) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.
3. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of the products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. PoLabs assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
4. PoLabs has used reasonable care in preparing the information included in this document, but PoLabs does not warrant that such information is error free. PoLabs assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
5. PoLabs devices may be used in equipment that does not impose a threat to human life in case of the malfunctioning, such as: computer interfaces, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment, and industrial robots.
6. Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when PoLabs devices are used for or in connection with equipment that requires higher reliability, for example: traffic control systems, anti-disaster systems, anticrime systems, safety equipment, medical equipment not specifically designed for life support, and other similar applications.
7. PoLabs devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety, as for example: aircraft systems, aerospace equipment, nuclear reactor control systems, medical equipment or systems for life support (e.g. artificial life support devices or systems), and any other applications or purposes that pose a direct threat to human life.
8. You should use the PoLabs products described in this document within the range specified by PoLabs, especially with respect to the maximum rating, operating supply voltage range and other product characteristics. PoLabs shall have no liability for malfunctions or damages arising out of the use of PoLabs products beyond such specified ranges.
9. Although PoLabs endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, PoLabs products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a PoLabs product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures.
10. Usage: the software in this release is for use only with PoLabs products or with data collected using PoLabs products.
11. Fitness for purpose: no two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.
12. Viruses: this software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.
13. Upgrades: we provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.
14. Please contact a PoLabs support for details as to environmental matters such as the environmental compatibility of each PoLabs product. Please use PoLabs products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. PoLabs assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
15. Please contact a PoLabs support at support@poscope.com if you have any questions regarding the information contained in this document or PoLabs products, or if you have any other inquiries.
16. The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.
17. Trademarks: Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56U, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.

Contents

1. Description 8
2. Features 9
2.1. PoKeys55 9
2.2. PoKeys56U 9
2.3. PoKeys56E 10
3. Device comparison 11
4. Device pin description table 12
4.1. PoKeys56U/E pin functions graphical overview 16
4.2. PoKeys55 pin functions graphical overview (serial numbers above 11500) 16
5. Requirements 17
5.1. USB devices 17
5.2. Network devices 17
6. Technical specifications 18
6.1. PoKeys55 and PoKeys56U dimensions 18
6.2. PoKeys56E dimensions 19
6.3. Electrical specification – limiting values 20
6.4. Electrical specification – static characteristic 20
6.5. Power supply for PoKeys56E 20
7. Installation 21
8. PoKeys configuration options 22
8.1. Digital inputs and outputs 22
PoKeys configuration software usage 22
Applicable PoKeys library commands 27
8.2. Connection signal output 28
PoKeys configuration software usage 29
Applicable PoKeys library commands 29
8.3. Digital counters (only on PoKeys56 devices) 30
PoKeys configuration software usage 30
Applicable PoKeys library commands 30
8.4. Encoders 32
PoKeys configuration software usage 32
Applicable PoKeys library commands 33
8.5. Matrix keyboard 35

PoKeys configuration software usage 35
Applicable PoKeys library commands 37
8.6. Analog inputs 39
PoKeys configuration software usage 39
Applicable PoKeys library commands 40
8.7. Joystick mapping 41
PoKeys configuration software usage 41
Applicable PoKeys library commands 42
8.8. PWM outputs 43
PoKeys configuration software usage 43
Applicable PoKeys library commands 44
8.9. Matrix LED output 45
PoKeys configuration software usage 46
Applicable PoKeys library commands 46
8.10. LCD 47
PoKeys configuration software usage 48
Applicable PoKeys library commands 49
8.11. Pulse engine 50
PoKeys device pins in use 50
Pulse engine operating principles 50
Safety charge-pump output 52
Axis parameters 52
PoKeys configuration software usage 52
Applicable PoKeys library commands 52
8.12. PoExtBus 54
PoKeys configuration software usage 55
Applicable PoKeys library commands 55
8.13. PoNET 56
Adding new devices 56
PoNET kb48CNC keyboard 57
PoNET/PoExtBus connection example 58
Applicable PoKeys library commands 58
8.14. Failsafe settings 60
Applicable PoKeys library commands 60

8.15. I²C protocol 61
Applicable PoKeys library commands 61
8.16. 1-wire 63
Applicable PoKeys library commands 63
8.17. Sensors 64
I²C sensors 64
1-wire sensors 64
Analog sensors 65
Applicable PoKeys library commands 66
8.18. PoKeys56E 67
Connecting to PoKeys56E device for the first time 67
Device discovery 68
Default settings 68
Connecting to device in other network 68
Security 68
Web interface 70
Modbus 73
Connecting to PoKeys56E across the internet 75
Using PoKeys software with PoKeys56E device across the internet 75
8.19. http://cosm.com/ service support 77
8.20. Changing User ID number 81
8.21. Saving current configuration to file 81
9. PoTLog27 firmware 81
10. Communicating with the USB devices using the console 85
10.1. Supported operations 85
11. Connecting common peripherals to PoKeys devices 88
12. Quick resetting the device configuration (or recovering bad firmware update) 94
13. Frequently asked questions 95
14. PoKeys library functions 98
14.1. EnumerateDevices (for USB-based PoKeys devices) 98
14.2. ConnectToDevice (for USB-based PoKeys devices) 98
14.3. DisconnectDevice 98
14.4. StartEthernetDiscovery 99
14.5. StopEthernetDiscovery 99

14.6. GetNumberOfDetectedNetworkDevices 99
14.7. ConnectToNetworkDevice 100
14.8. GetDeviceID 100
14.9. GetDeviceIDEx 100
14.10. GetBuildDate 101
14.11. GetUserID 101
14.12. SetUserID 101
14.13. SetPinData 102
14.14. SetPinData 102
14.15. GetPinData 103
14.16. GetPinData 104
14.17. GetInput 104
14.18. SetOutput 105
14.19. GetAnalogInput 105
14.20. SetAnalogOutput 106
14.21. SaveConfiguration 107
14.22. GetMatrixConfiguration 107
14.23. SetMatrixConfiguration 108
14.24. GetPWMOutputs 110
14.25. SetPWMOutputs 111
14.26. LCDSetSettings 111
14.27. LCDInit 112
14.28. LCDClear 112
14.29. LCDGotoXY 113
14.30. LCDPutc 113
14.31. LCDPrint 113
14.32. LCDSetEntryMode 114
14.33. LCDDisplayOnOffControl 114
14.34. LCDDefineCustomCharacter 115
14.35. AuxilaryBusGetData 115
14.36. AuxilaryBusSetData 115
15. Interfacing with PoKeys library – C# example 117
16. Major changes from 1.x to 1.7: 126
16.1. Pin 13 not functioning appropriately 126

16.2. Putting pin 4 low on startup disables PoKeys55 device from booting. 126
17. Errata information 127
Pins 5 and 6 cannot be separately set as outputs/inputs 127
18. Grant of license 128

1. Description

PoKeys products line consists of simple, easy-to-use USB and network devices with the extended list of features making them powerful input/output devices. PoKeys55 and PoKeys56U devices also incorporate a virtual USB keyboard and joystick with a simple setup. PoKeys devices enable user to design specially built robust computer interfaces. The devices are highly adjustable and as such require no complex knowledge on device programming. Virtual USB keyboard supports emulation of single key presses or various programmable series of keys (keyboard macro sequences) while virtual USB joystick emulation supports mapping of analog inputs to joystick axes and mapping of digital inputs to joystick buttons.

If additional input and output capabilities are needed, the devices provide 55 digital 5V tolerant inputs or outputs, 5 10-bit analog inputs (on PoKeys55) / 7 12-bit analog inputs (on PoKeys56U/PoKeys56E) with adjustable software low-pass filter and one 10-bit analog output (on PoKeys55). These are complemented with 6 high-speed fully configurable PWM (pulse width modulation) outputs. User can freely set PWM period and PWM duty cycles. PWM module runs at 12 MHz (on PoKeys55) / 25 MHz (on PoKeys56U/PoKeys56E) and allows high-speed output switching.

They are controlled via included software, which enables user either to use the highly intuitive graphical user interface. Chosen settings can be stored on device, so no special software is needed on target system.

PoKeys devices also support up to 25 pairs of quadrature encoder signal inputs for which can be freely connected to any of the 55 inputs on PoKeys devices. These inputs increment or decrement the counters that can be read via provided software of other third-party applications. Changes in those signals can trigger virtual keyboard presses on USB versions of PoKeys devices. PoKeys devices also support three additional fast encoders, when higher frequencies of the quadrature encoder signals are in use. PoKeys56U and PoKeys56E add additional support for one ultra-fast quadrature encoder signal that can handle even greater frequencies of these signals with ease.

In the cases when the application requires more than 55 inputs or outputs, PoKeys devices have the inbuilt support for matrix keyboards sized up to 16x8 with freely assignable key codes for virtual USB keyboard, while the PoExtBus bus feature adds the support for additional 80 digital inputs.

PoKeys devices also support hd44780-compatible character LCD displays and two serially driven 8x8 matrix LED displays.

Third-party application developers that are adding the support for PoKeys devices, are encouraged to use the supplied communication DLL that can be simply used in the different .NET framework based applications and various other programming languages that provide support for ActiveX interface.

To aid developers that are communicating with PoKeys devices on the low-level, the extensive documentation on device communication protocol can be downloaded free of charge from the product webpage.

2. Features

2.1. PoKeys55

- Compatible with USB 1.1/2.0 HID standard
- Standard USB keyboard simulation (with triggering support for up/down keys)
- Standard USB joystick simulation (6 axis, 32 buttons with triggering support)
- 55 digital inputs with pull-up resistors, freely mappable to virtual USB keyboard's keys
- 55 software controlled digital outputs
- 5 analog inputs (10-bit), freely mappable to any of virtual USB joystick axes (with adjustable low-pass filtering support)
- 1 software controlled 10-bit analog output, controlled via included software
- Up to 25 encoder pair inputs
- Up to 64 256-character long keyboard macro sequences
- Up to 16x8 matrix keyboard with triggered keys/alternate function support
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (12 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- Support for Connection signal output
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

2.2. PoKeys56U

- Compatible with USB 1.1/2.0 HID standard
- Standard USB keyboard simulation (with triggering support for up/down keys)
- Standard USB joystick simulation (6 axis, 32 buttons with triggering support)
- 55 digital inputs with pull-up resistors, freely mappable to virtual USB keyboard's keys
- 55 software controlled digital outputs
- 7 analog inputs (12-bit) with adjustable low-pass filtering support
- Up to 26 encoder pair inputs (3 high-speed encoder inputs, 1 ultra high speed encoder input)
- Digital counters on specific digital input pins
- 3-axis 25 kHz Pulse engine with safety 5 kHz charge-pump output
- Two 8x8 matrix LED display support
- Up to 64 256-character long keyboard macro sequences
- Up to 16x8 matrix keyboard with triggered keys/alternate function support
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (25MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- PoNET devices support (48-key CNC keyboard mapped to matrix keyboard)
- Support for Connection signal output
- Fail-safe support in case of communication interruption
- Support for up to 10 sensors on I^2C bus, up to 10 sensors on 1-wire bus and up to 7 analog sensors

- Communication compatible with PoKeys55 on application layer (data packet structure)
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

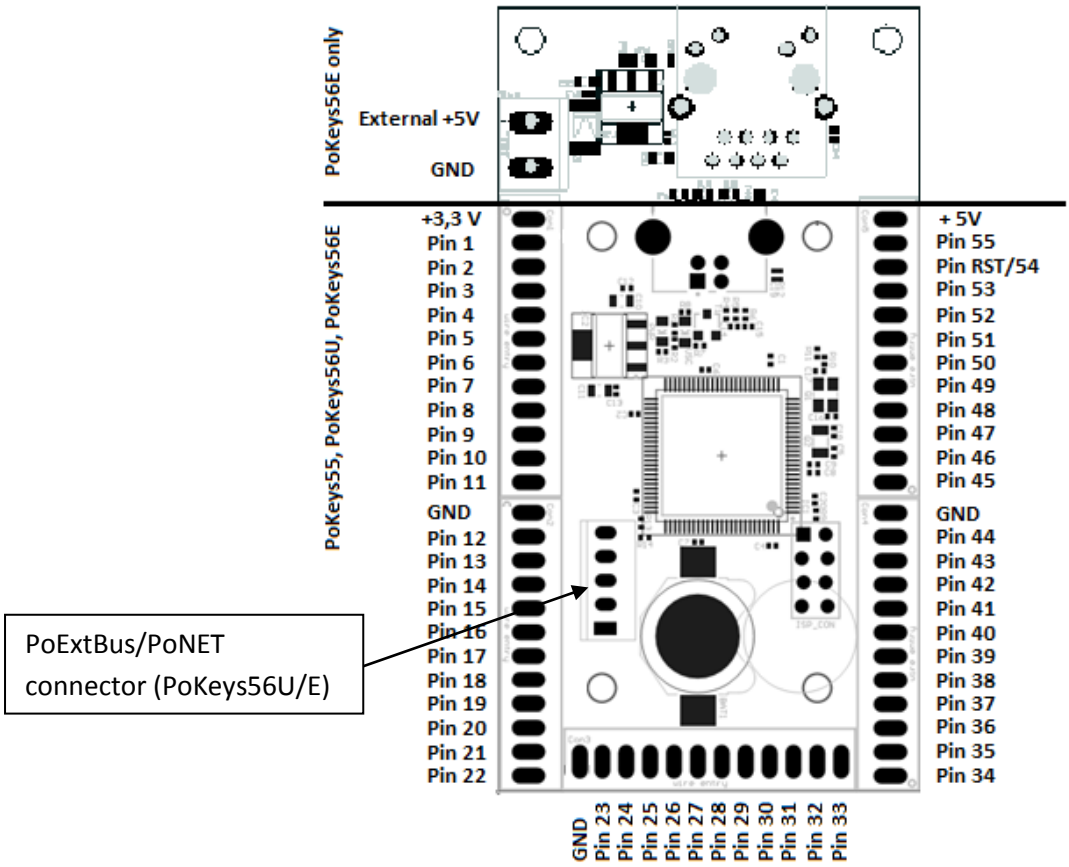
2.3. PoKeys56E

- Ethernet 10/100 with DHCP client or fixed IP support
- TCP connection with device
- 55 digital inputs with pull-up resistors
- 55 software controlled digital outputs
- 7 analog inputs (12-bit) with adjustable low-pass filtering support
- Up to 26 encoder pair inputs (3 high-speed encoder inputs, 1 ultra high speed encoder input)
- 3-axis 25 kHz Pulse engine with safety 5 kHz charge-pump output
- Digital counters on specific digital input pins
- Up to 16x8 matrix keyboard
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (25 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- PoNET devices support (48-key CNC keyboard)
- Modbus TCP support (access to digital IO, analog inputs, encoders' counters, digital counters values, PWM outputs, LCD display, LED matrix display, PoExtBus devices, matrix keyboard status)
- Support for up to 10 sensors on I^2C bus, up to 10 sensors on 1-wire bus and up to 7 analog sensors
- Web interface with newly designed dashboard and I/O status display with multiple user accounts
- Support for communication with devices on I^2C and 1-wire buses
- Support for Connection signal output
- Fail-safe support in case of communication interruption
- Communication compatible with PoKeys55 on application layer (data packet structure)
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

3. Device comparison

	PoKeys55	PoKeys56U	PoKeys56E
Number of pins on the board	55	55	55
Digital inputs	55	55	55
Digital outputs	55	55	55
Analog inputs	5x 10-bit	7x 12-bit	7x 12-bit
Analog ouputs	1x 10-bit	0	0
Number of encoders (normal)	25	25	25
Number of fast encoders	3	3	3
Number of ultra fast encoders	0	1	1
Number of PWM outputs	6 (12 MHz clock)	6 (25 MHz clock)	6 (25 MHz clock)
Number of digital counters	0	24	24
Pulse engine	not suported	3-axis 25 kHz	3-axis 25 kHz
LCD support	Alphanumeric up to 4x20	Alphanumeric up to 4x20	Alphanumeric up to 4x20
Matrix keyboard support	Up to 16x8	Up to 16x8	Up to 16x8
Matrix LED	Two 8x8	Two 8x8	Two 8x8
Keyboard emulation	Yes	Yes	No
Joystick emulation	Yes	Yes	No
Connectivity	USB	USB	Ethernet (IPv4 + DHCP)
Modbus TCP support	No	No	Yes
I2C sensors	No	Yes	Yes
1-wire sensors	No	Yes	Yes
PoExtBus support	up to 10 devices	up to 10 devices	up to 10 devices
PoNET bus support	No	up to 16 devices	up to 16 devices
Power supply	powered from USB	powered from USB	external 5V power supply
Number of devices per computer	127 per USB root hub	127 per USB root hub	practically unlimited
Automatic device discovery	Yes	Yes	Yes (in local network)
Free configuration application	Yes	Yes	Yes
Free firmware upgrades	Yes	Yes	Yes
Communication DLL support	Yes	Yes	Yes
Availability	Out of production	Available	Available

4. Device hardware description



Pin	Type	Description
1	I/O	General purpose digital input/output pin
	I	**Fast encoder 1A** – A channel input for the fast encoder 1
	I	**Counter 1** – Counter input signal for counter 1 (not available on PoKeys55)
2	I/O	General purpose digital input/output pin
	I	**Fast encoder 1B** – B channel input for the fast encoder 1
	I	**Counter 2** – Counter input signal for counter 2 (not available on PoKeys55)
3	I/O	General purpose digital input/output pin
4	I/O	General purpose digital input/output pin
5	I/O	General purpose digital input/output pin (when used as output on PoKeys56E, pin 6 must also be specified as output)
	I	**Fast encoder 2A** – A channel input for the fast encoder 2
	I	**Counter 5** – Counter input signal for counter 5 (not available on PoKeys55)
6	I/O	General purpose digital input/output pin (when used as output on PoKeys56E, pin 5 must also be specified as output)
	I	**Fast encoder 2B** – B channel input for the fast encoder 2
	I	**Counter 6** – Counter input signal for counter 6 (not available on PoKeys55)

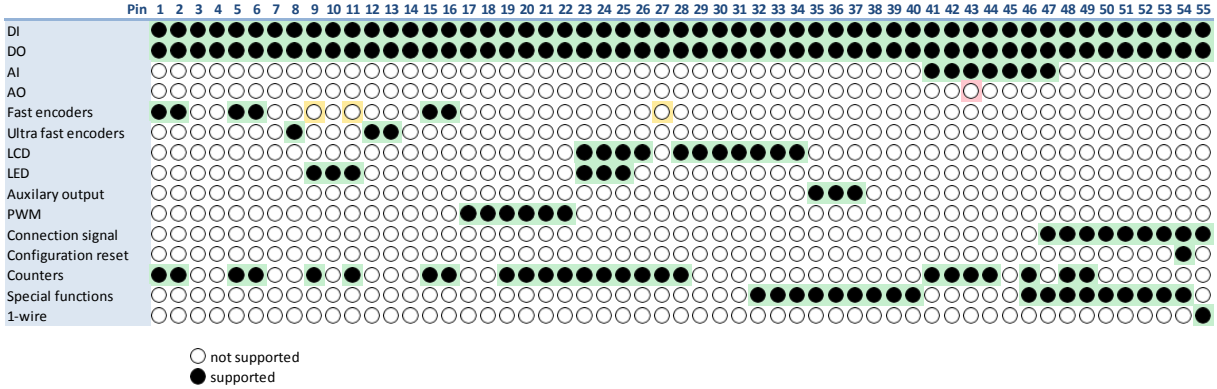
7	I/O	General purpose digital input/output pin
8	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder input A** (not available on PoKeys55)
9	I/O	General purpose digital input/output pin
	O	**Matrix LED 1** serial data output
	I	**Counter 9** – Counter input signal for counter 9 (not available on PoKeys55)
10	I/O	General purpose digital input/output pin
	O	**Matrix LED 1** output latch clock output
11	I/O	General purpose digital input/output pin
	O	**Matrix LED 1** serial clock output
	I	**Counter 11** – Counter input signal for counter 11 (not available on PoKeys55)
12	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder input B** (not available on PoKeys55)
13	I/O	General purpose digital input/output pin
	I	**Ultra fast encoder index** signal input (not available on PoKeys55)
14	I/O	General purpose digital input/output pin
15	I/O	General purpose digital input/output pin
	I	**Fast encoder 3A** – A channel input for the fast encoder 3
	I	**Counter 15** – Counter input signal for counter 15 (not available on PoKeys55)
16	I/O	General purpose digital input/output pin
	I	**Fast encoder 3B** – B channel input for the fast encoder 3
	I	**Counter 16** – Counter input signal for counter 16 (not available on PoKeys55)
17	I/O	General purpose digital input/output pin
	O	**PWM channel 6** output
18	I/O	General purpose digital input/output pin
	O	**PWM channel 5** output
19	I/O	General purpose digital input/output pin
	O	**PWM channel 4** output
	I	**Counter 19** – Counter input signal for counter 19 (not available on PoKeys55)
20	I/O	General purpose digital input/output pin
	O	**PWM channel 3** output
	I	Counter 20 – Counter input signal for counter 20 (not available on PoKeys55)
21	I/O	General purpose digital input/output pin
	O	**PWM channel 2** output
	I	**Counter 21** – Counter input signal for counter 21 (not available on PoKeys55)
22	I/O	General purpose digital input/output pin
	O	**PWM channel 1** output
	I	**Counter 22** – Counter input signal for counter 22 (not available on PoKeys55)
23	I/O	General purpose digital input/output pin
	I/O	**LCD data 7** (primary configuration)
	O	**Matrix LED 2** serial data output
	I	**Counter 23** – Counter input signal for counter 23 (not available on PoKeys55)
24	I/O	General purpose digital input/output pin
	I/O	**LCD data 6** (primary configuration)
	O	**Matrix LED 2** output latch clock output
	I	**Counter 24** – Counter input signal for counter 24 (not available on PoKeys55)

25	I/O	General purpose digital input/output pin
	I/O	**LCD data 5** (primary configuration)
	O	**Matrix LED 2** serial clock output
	I	**Counter 25** – Counter input signal for counter 25 (not available on PoKeys55)
26	I/O	General purpose digital input/output pin
	I/O	**LCD data 4** (primary configuration)
	I	**Counter 26** – Counter input signal for counter 26 (not available on PoKeys55)
27	I/O	General purpose digital input/output pin
	I	**Counter 27** – Counter input signal for counter 27 (not available on PoKeys55)
28	I/O	General purpose digital input/output pin
	O	**LCD R/W** (primary and secondary configuration)
	I	**Counter 28** – Counter input signal for counter 28 (not available on PoKeys55)
29	I/O	General purpose digital input/output pin
	O	**LCD E** (primary and secondary configuration)
30	I/O	General purpose digital input/output pin
	O	**LCD RS** (primary and secondary configuration)
31	I/O	General purpose digital input/output pin
	I/O	**LCD data 7** (secondary configuration)
32	I/O	General purpose digital input/output pin
	I/O	**LCD data 6** (secondary configuration)
	I	Pulse engine **LIMIT+ x** switch input - x-axis limit switch for the positive direction (not available on PoKeys55)
33	I/O	General purpose digital input/output pin
	I/O	**LCD data 5** (secondary configuration)
	I	Pulse engine **LIMIT+ y** switch input - y-axis limit switch for the positive direction (not available on PoKeys55)
34	I/O	General purpose digital input/output pin
	I/O	**LCD data 4** (secondary configuration)
	I	Pulse engine **LIMIT+ z** switch input - z-axis limit switch for the positive direction (not available on PoKeys55)
35	I/O	General purpose digital input/output pin
	O	**PoExtBus** serial clock if PoExtBus on dedicated connector is not used
	I	Pulse engine **LIMIT- x** switch input - x-axis limit switch for the negative direction (not available on PoKeys55)
36	I/O	General purpose digital input/output pin
	O	**PoExtBus** serial data if PoExtBus on dedicated connector is not used
	I	Pulse engine **LIMIT- y** switch input - y-axis limit switch for the negative direction (not available on PoKeys55)
37	I/O	General purpose digital input/output pin
	O	**PoExtBus** output latch clock if PoExtBus on dedicated connector is not used
	I	Pulse engine **LIMIT- z** switch input - z-axis limit switch for the negative direction (not available on PoKeys55)
38	I/O	General purpose digital input/output pin
	O	Pulse engine **DIR x** - x-axis direction output (not available on PoKeys55)
39	I/O	General purpose digital input/output pin
	O	Pulse engine **DIR y** - y-axis direction output (not available on PoKeys55)
40	I/O	General purpose digital input/output pin

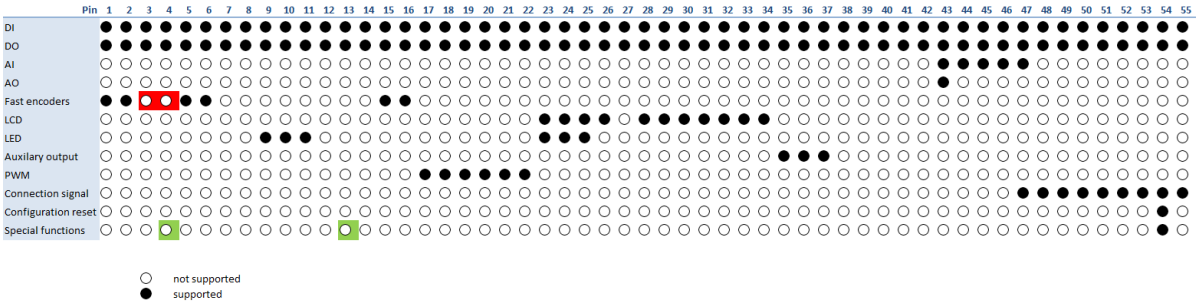
	O	Pulse engine **DIR z** - z-axis direction output (not available on PoKeys55)
41	I/O	General purpose digital input/output pin
	I	**Analog input 41** (not available on PoKeys55)
	I	**Counter 41** – Counter input signal for counter 41 (not available on PoKeys55)
42	I/O	General purpose digital input/output pin
	I	**Analog input 42** (not available on PoKeys55)
	I	**Counter 42** – Counter input signal for counter 42 (not available on PoKeys55)
43	I/O	General purpose digital input/output pin
	I	**Analog input 43**
	I	**Counter 43** – Counter input signal for counter 43 (not available on PoKeys55)
44	I/O	General purpose digital input/output pin
	I	**Analog input 44**
	I	**Counter 44** – Counter input signal for counter 44 (not available on PoKeys55)
45	I/O	General purpose digital input/output pin
	I	**Analog input 45**
46	I/O	General purpose digital input/output pin
	I	**Analog input 46**
	I	**Counter 46** – Counter input signal for counter 46 (not available on PoKeys55)
	O	Pulse engine **STEP x** - x-axis step output (not available on PoKeys55)
47	I/O	General purpose digital input/output pin
	I	**Analog input 47**
	I	Pulse engine **HOME x** - x-axis home switch input (not available on PoKeys55)
	O	Connection signal output
48	I/O	General purpose digital input/output pin
	I	**Counter 48** – Counter input signal for counter 48 (not available on PoKeys55)
	O	Pulse engine **STEP y** - y-axis step output (requires additional 470 Ohm pull-up resistor on PoKeys56U) (not available on PoKeys55)
	O	Connection signal output
49	I/O	General purpose digital input/output pin
	I	**Counter 49** – Counter input signal for counter 49 (not available on PoKeys55)
	O	Pulse engine **STEP z** - z-axis step output (requires additional 470 Ohm pull-up resistor on PoKeys56U) (not available on PoKeys55)
	O	Connection signal output
50	I/O	General purpose digital input/output pin
	I	Pulse engine **HOME y** - y-axis home switch input (not available on PoKeys55)
	O	Connection signal output
51	I/O	General purpose digital input/output pin
	I	Pulse engine **HOME z** - z-axis home switch input (not available on PoKeys55)
	O	Connection signal output
52	I/O	General purpose digital input/output pin
	I	Pulse engine **EMGERGENCY** - emergency switch input (not available on PoKeys55)
	O	Connection signal output
53	I/O	General purpose digital input/output pin
	O	Pulse engine **CHARGEPUMP** - 5 kHz charge pump signal output (not available on PoKeys55)
	O	Connection signal output

54	I/O	General purpose digital input/output pin
	I	Recovery mode (**RST**) signal - PoKeys enters recovery mode if this signal is tied to GND at startup
	O	Connection signal output
55	I/O	General purpose digital input/output pin
	I/O	**1-wire** communication bus (not available on PoKeys55)
	O	Connection signal output

4.1. PoKeys56U/E pin functions graphical overview


Pin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
DI
DO
AI
AO
Fast encoders
Ultra fast encoders
LCD
LED
Auxilary output
PWM
Connection signal
Configuration reset
Counters
Special functions
1-wire
not supported
supported

4.2. PoKeys55 pin functions graphical overview **(serial numbers above 11500)**


Pin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
DI
DO
AI
AO
Fast encoders
LCD
LED
Auxilary output
PWM
Connection signal
Configuration reset
Special functions
not supported
supported

4.3. Status LEDs

Each PoKeys devices has the following status LEDs:

- Red LED – power status: when the power is applied to PoKeys device, this LED is lit
- Green LED – connection/communication status: this LED reflects various operating modes of the PoKeys device
    - Fast blinking on startup: device has started in the recovery mode. This enables to update the firmware and clear the settings in the device.
    - Slow blinking on startup (PoKeys56E): PoKeys device is waiting for DHCP settings
    - Solid: device is in normal operation mode

5. Requirements

5.1. USB devices

- One available USB 1.1 or USB 2.0 port
- USB HID device driver enabled operating system (Windows 98 SE/ME/2000/XP/Vista, Linux, Mac OS)
- Included software requires Windows 2000/XP/Vista/Windows 7 with .NET framework 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD USB KEYBOARD AND JOYSTICK).

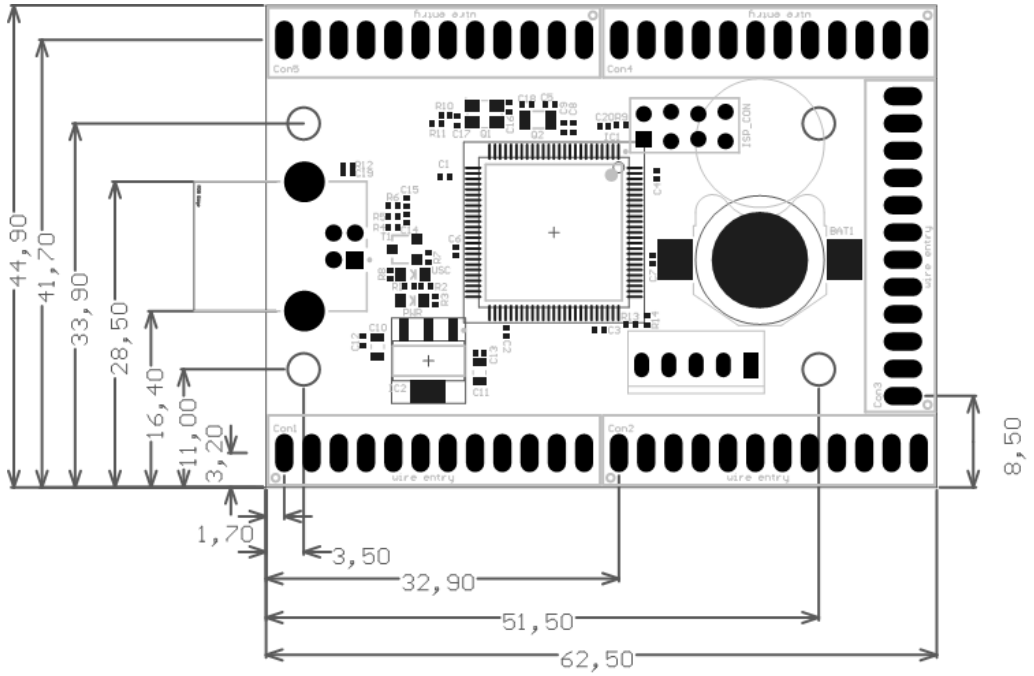
5.2. Network devices

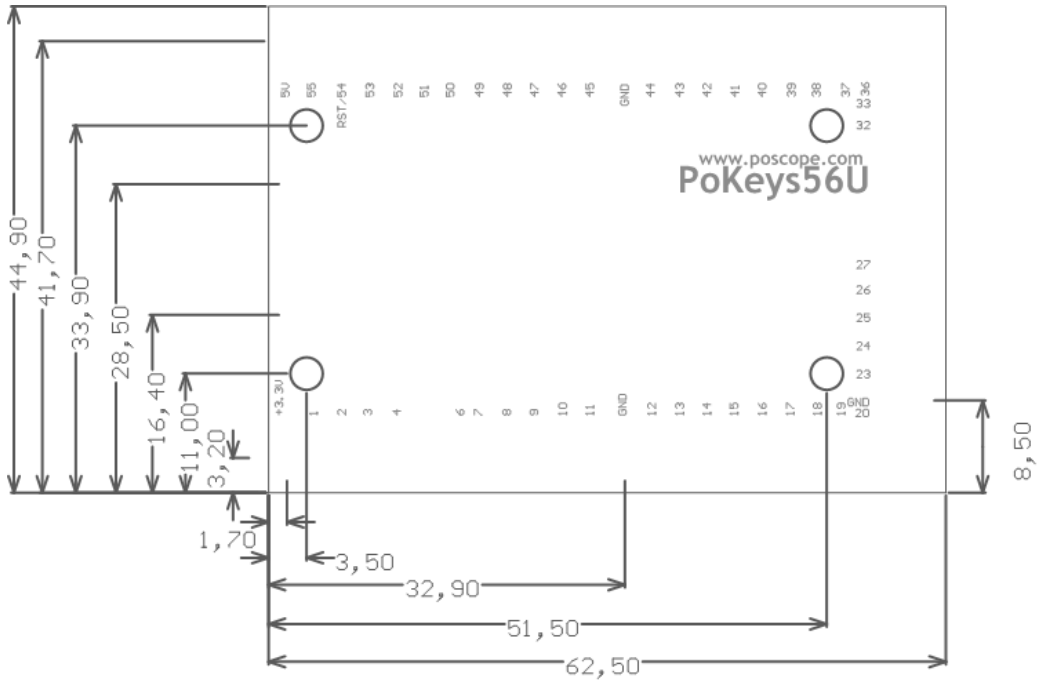
- Ethernet connection between host computer and PoKeys56E device
- 5V DC power supply with 400 mA. If any additional peripheral is connected to PoKeys56E, use appropriately more powerful power supply.
- Included software requires Windows 2000/XP/Vista/Windows 7 with .NET framework 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD NETWORK DEVICE).

6. Technical specifications

6.1. PoKeys55 and PoKeys56U dimensions

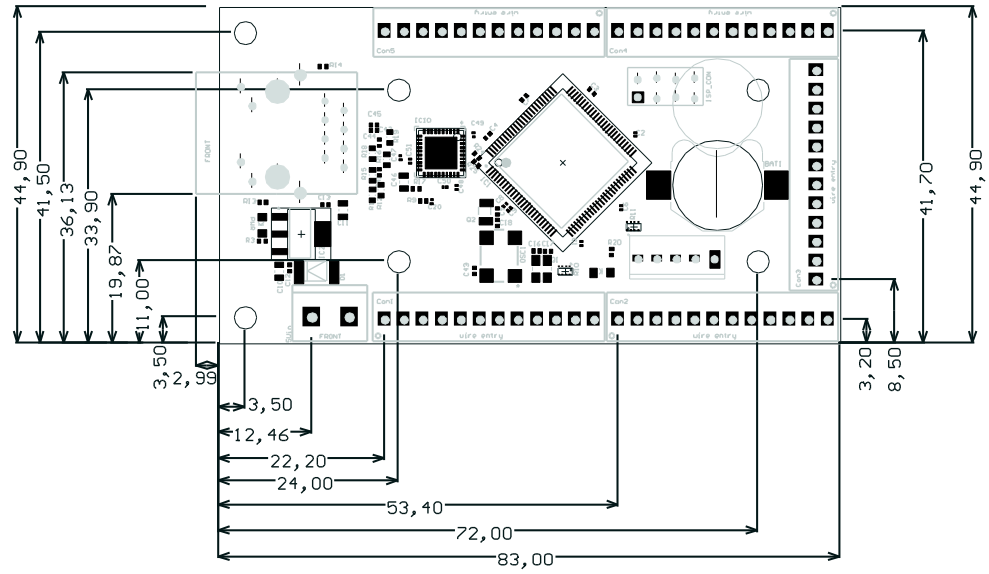
Measurements are in mm, all holes have 3mm diameter.

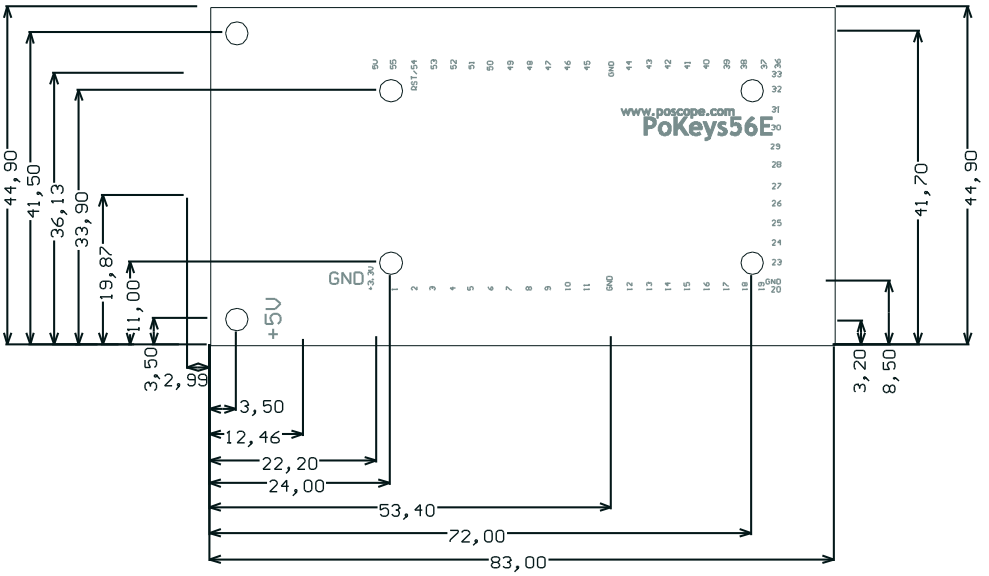




6.2. PoKeys56E dimensions

Measurements are in mm, all holes have 3mm diameter.





6.3. Electrical specification – limiting values

Symbol	Parameter	Min	Max	Unit
V_{IA}	analog input voltage on ADC related pins	-0.5	5.1	V
V_I	Input voltage on other pins	-0.5	6.0	V
V_{esd}	electrostatic discharge	-4000	4000	V

6.4. Electrical specification – static characteristic

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_I	input voltage	pin configured as digital input	0	-	5.5	V
V_O	output voltage	pin configured as digital output	0	-	3.3	V
V_{IH}	HIGH-level input voltage		2.0	-	-	V
V_{I_analog}	analog input voltage	pin configured as analog input	0	-	3.3	V
V_{O_analog}	analog output voltage	pin configured as analog output	0	-	3.3	V
V_{IL}	LOW-level input voltage		-	-	0.8	V
V_{hys}	hysteresis voltage		0.4	-	-	V
V_{OH}	HIGH-level output voltage	I_{OH} = -4 mA	2.9	-	-	V
V_{OL}	LOW-level output voltage	I_{OH} = 4 mA	-	-	0.4	V
I_{OH}	HIGH-level output current	V_{OH} = 2.9 V	-4	-	-	mA
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	4	-	-	mA
I_{OHS}	HIGH-level short-circuit output current	V_{OH} = 0 V	-	-	-45	mA
I_{OLS}	LOW-level short-circuit output current	V_{OL} = 3.3V	-	-	50	mA
I_{pu}	pull-up current		0 (V_I >= 3.3V)	-50	-100	µA
I_{5V}	5V pin source current				400	mA
I_{5V}	5V power supply current for PoKeys56E **without peripherals**		250	300	400	mA

6.5. Power supply for PoKeys56E

PoKeys56E requires external 5V power supply to be connected to the board. If any additional peripheral is connected to PoKeys56E board, combined current of peripheral and PoKeys56E board must be taken into consideration when selecting the power supply.

Due to the protection diode in the circuit, a 0.8 V voltage drop can be observed between input voltage and 5V pin. When supplying the PoKeys56E board with 5V power supply, take care when connecting peripherals that require 5V supply voltage (e.g., LCD displays).

7. Installation

PoKeys55 and PoKeys56U are USB 1.1/2.0 compliant devices and as such require no additional drivers for operation as a standard USB keyboard and joystick.

PoKeys56E is an Ethernet device that requires an Ethernet connection between host computer and the device and 5V power supply. No additional drivers are required. Network firewalls must allow all traffic on TCP/UDP port 20055.

To operate the device after the device has been configured there is no software installation necessary on a target system.

To configure the device the supplied software must be installed and the requirements listed in previous section of this manual must be met.

8. PoKeys configuration options

8.1. Digital inputs and outputs

Any of the 55 pins, available on the terminals on the PoKeys PCB, can be configured as digital input or output with selectable polarity. All pins have a weak pull-up resistor enabled and are 5V tollerant.

At every startup, all digital input or output configured pins are preset as digital input pins (with the weak pull-up resistor pulling the state of the pin to logical 1). To enable output function on a selected pin, SetPinFunction command must be executed. This behaviour can be disabled using the SetAutoSetOutputs command or via PoKeys configuration software (Settings > Initialize outputs on startup). Each output pin can sink or source up to 4 mA of current, with the limitation that the pins combined source or sink current does not exceed 100 mA.

All digital input pins on PoKeys55 or PoKeys56U can be configured to simulate a configurable USB keyboard key. When there is a high state on pin (on low state when using inverted polarity option) PoKeys device sends a USB message with the key code and modifier associated with this pin. Moreover, PoKeys device can simulate a series of key presses, what is called a macro sequence. Up to 64 different macro sequences can be setup with the combined total length of 3584 characters with each macro sequence shorter than 128 keys. All macros can be labelled with a 7-character name. Character codes (most frequently used are listed in the Appendix to this document) are USB HID standard keyboard codes.

An extension to the USB keyboard mapping described above, PoKeys supports also triggered mapping of inputs to USB keyboard keys. In triggered mapping mode, only pin state transitions (low-to-high or high-to-low trigger a USB key press) with different key combinations for each transition.

Type-matic like repeat and delay is an additional extension to the triggered key mapping. Instead of relying on the user's system to trigger key repeat events, PoKeys can be configured to simulate repeated key presses at the predefined rate (period between two key presses is adjustable in 5 ms cycles – 0.78 to 200 repeats possible[1]) after a predefined delay (adjustable in steps of 5 ms – 0 to 1275 ms possible).

PoKeys configuration software usage

There is graphical representation for configuration of each PoKeys device's pin on left and right side of main window. To change pin function, click on pin name and change its function in central 'Pin settings' frame.

[1] The maximum repeat rate depends on the user's system

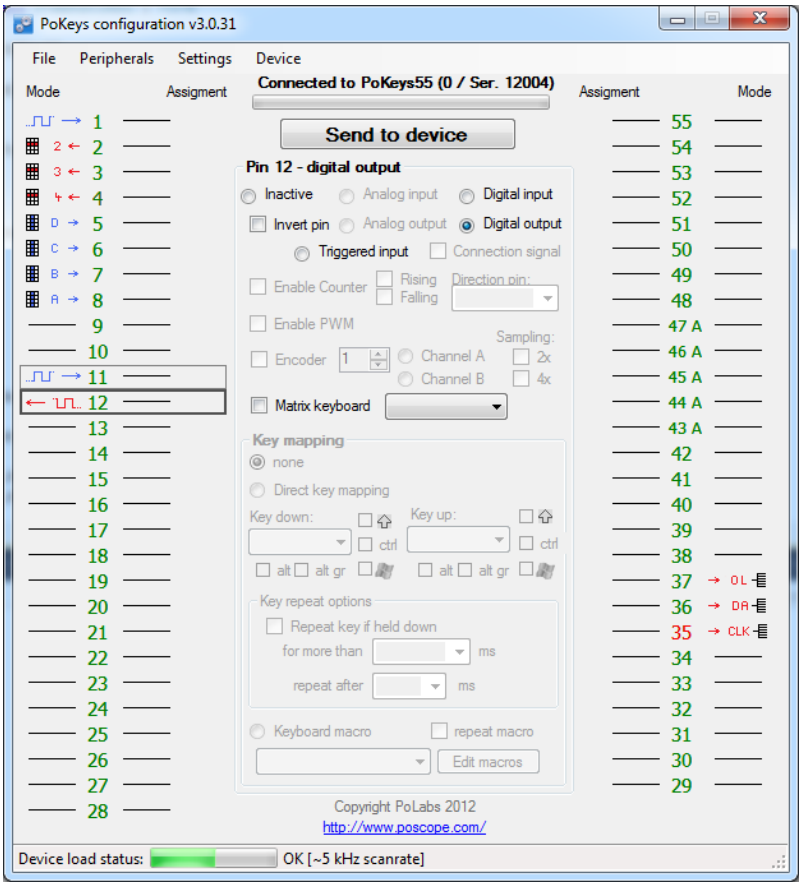


Figure 1: PoKeys configuration window

There are 6 main pin functions possible: inactive, digital input, triggered digital input, digital output, analog input and analog output.

Inactive

Any pin (except those fixly mapped to an activated peripheral) can be set as inactive. Inactive pin is put in high-Z state with internal pull-up resistors enabled.

Digital input

Any one of the 55 pins can be configured as digital input by selecting ‘Digital input’ option box. If the pin polarity is wished to be inverted, check the 'Invert pin' box.

There are several additional possibilities for digital input pin functions.

Direct key mapping – only for USB devices

Digital input set up for direct key mapping acts like a keyboard key. When there is a high state on pin (on low state when using inverted option), PoKeys55/PoKeys56U sends a key associated with this pin. Select a keyboard key from drop-down box and check appropriate key modifiers (Shift, Ctrl, …).

Example: Send Alt-F4

Select F4 from drop-down box and check Alt checkbox.

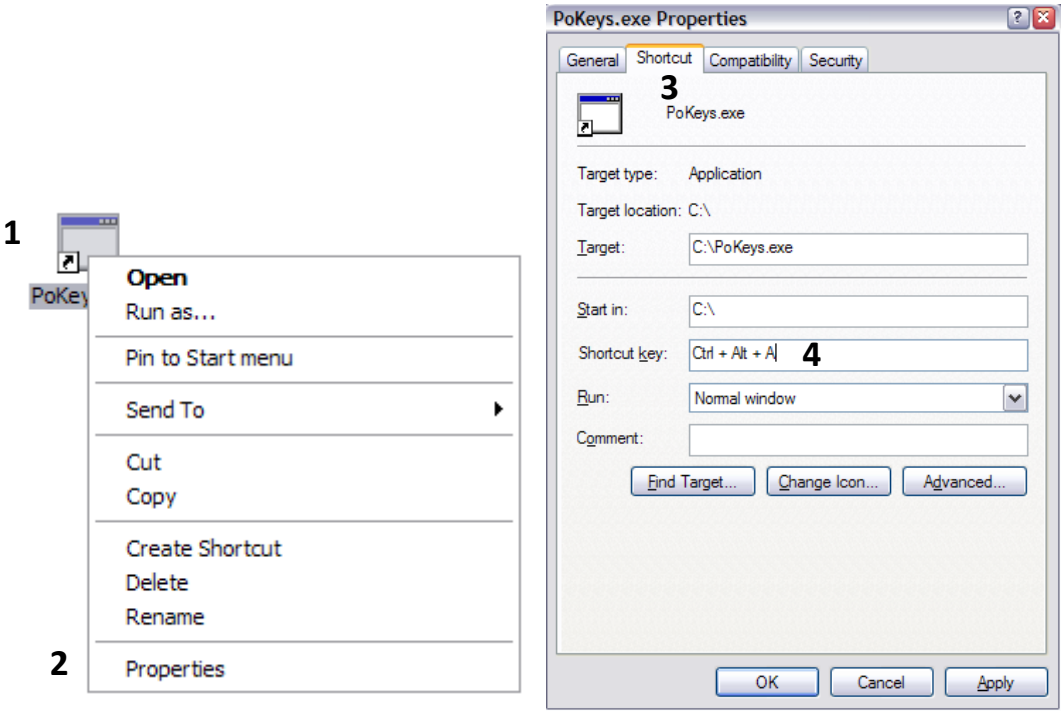
Example: Send ((opening bracket)

This key kombination differs from your system regional settings. As the PoKeys55/PoKeys56U devices emulate a system keyboard, key associations depend on current sytem keyboard regional setting. To

send an opening bracket symbol, one possible solution is to press Shift-8 (in most non-English countries) or to press Shift-9. Out of this reason there are no such secondary keys listed in drop-down box and must be entered by user as described above.

Example: starting a program on Windows using PoKeys55/PoKeys56U device

On a Windows operating system, users can assign a custom shortcut key to any program shortcut. Find the shortcut and then right click on it to show the context menu (Step 1). Select Properties (Step 2), and under the Shortcut tab (Step 3), click on the 'Shortcut key' text box. Proceed by typing in a combination that you wish to assign to a particular program (Step 4). Next, open the PoKeys application and connect to the desired PoKeys device. Click on the pin that will function as a launch trigger for your application (Step 5). Under Key mapping, select the same keyboard combination that you assigned to the program shortcut (Step 6). Click on the 'Send to device' button (Step 7) to transfer settings to the device. This will activate the new shortcut.



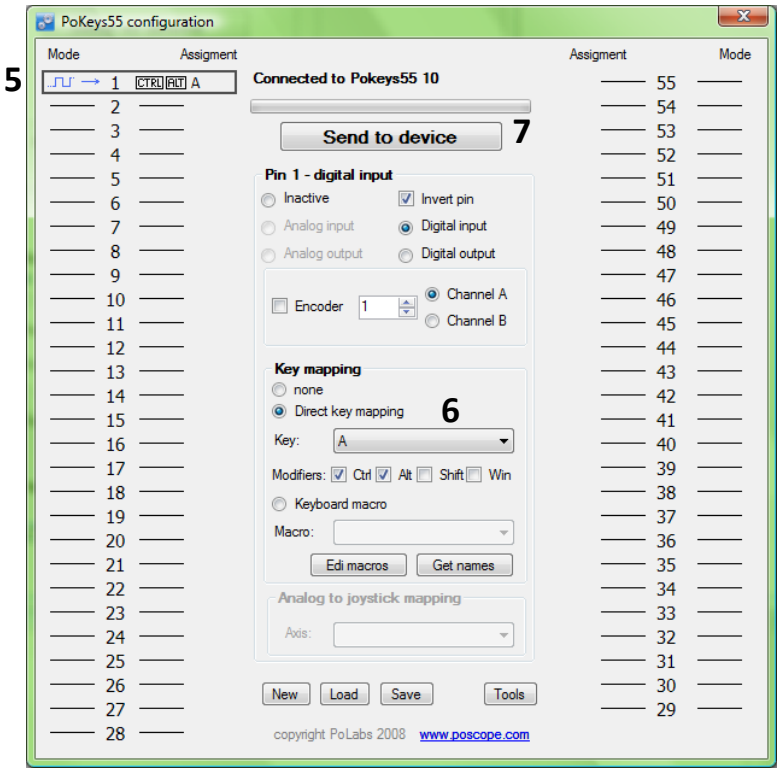


Figure 2: Setting up PoKeys device

Keyboard macro – only for USB devices

PoKeys55 and PoKeys56U devices support keyboard macros – the key press combinations that can be up to 256 keys long. To define a keyboard macro, first select Keyboard macro mapping option for one of the pins. 'Edit macros' and 'Get names' command buttons become enabled. To add, change or delete macros click the 'Edit macros' button. The following dialog appears

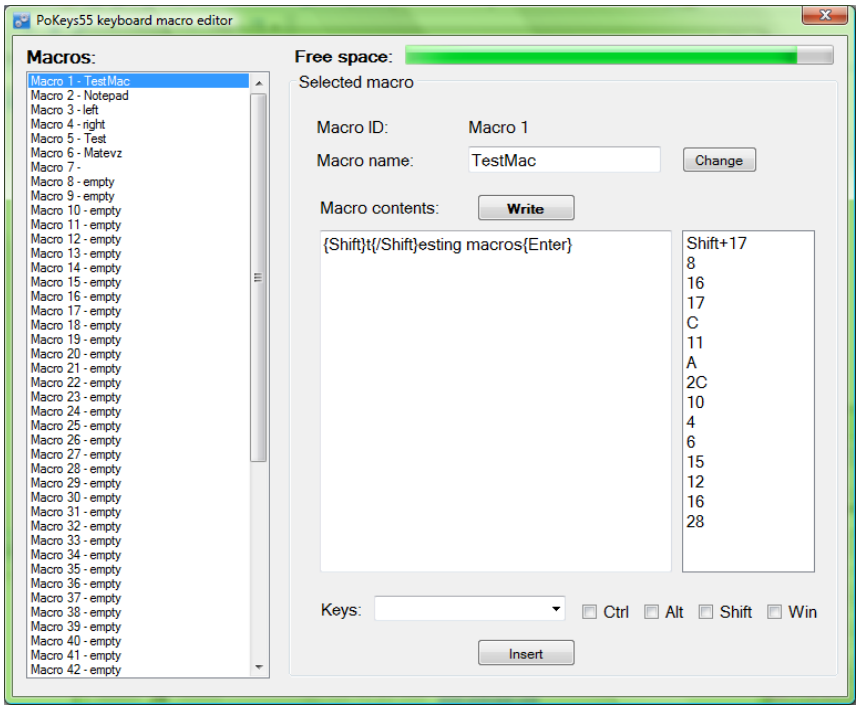


Figure 3: Macro editing dialog

First select the macro you want to edit. To change macro name, enter desired macro name (up to 7 characters long) in 'Macro name' text box and click 'Change' button. This name is used only to help user differentiate between multiple macros.

To set macro contents, simply enter text into 'Macro contents' text box. If there is an invalid character found, the text appears red. When finished, click Write to write macro to device.

List box at the right displays final macro sequence that is sent to PC when the macro gets activated.

Triggered digital input – only for USB devices

Any one of the 55 pins can be configured as triggered digital input by selecting 'Triggered input' option box. This pin mode enables user to select a key that is pressed only when a transition in a signal occurs. Different keys can be selected for 'LOW-to-HIGH' or 'Key up' event and for 'HIGH-to-LOW' or 'Key down' event. Selecting the keys is similar to Direct key mapping described above.

Digital output

Any one of the 55 pins can be configured as digital output by selecting 'Digital output' option box. If the polarity of the pin is wished to be inverted, check the 'Invert pin' box. On startup, all pins (although optionally configured as digital output) are by default initialized in high-Z state (behaving like inputs). To use the outputs, SetPinData should be called before attempting to set the output state. However, there is an option to disable this behaviour - use the 'Settings > Initialize outputs on startup' option to either enable or disable output activation on PoKeys startup.

View status of digital inputs and outputs

Go to 'Peripherals > Digital inputs and outputs…' to display the status dialog as shown below.

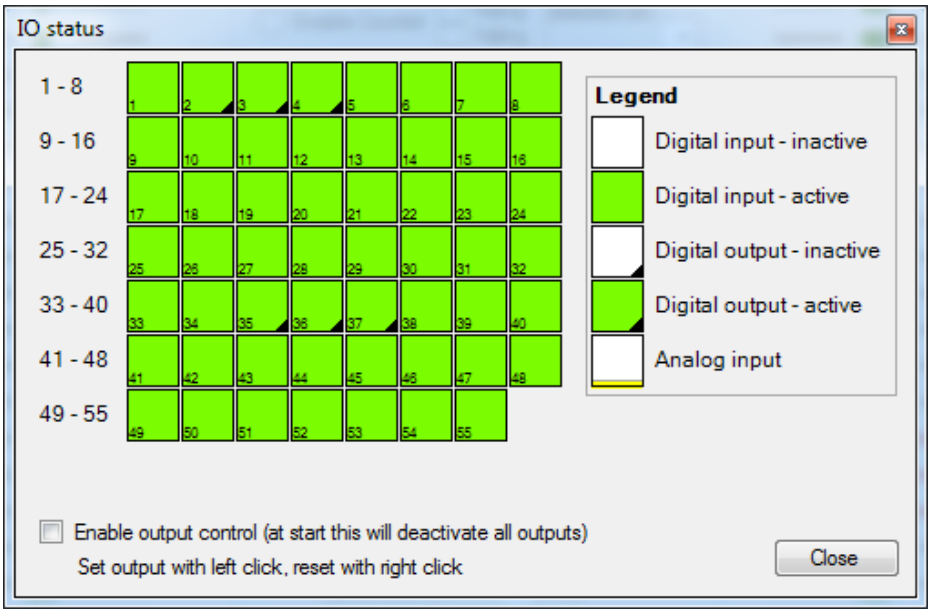


Figure 4: Input and output status dialog

There are 55 pins represented as colored squares in the dialog, organized in the rows of 8 pins. Each square contains a pin index number in the lower left corner, while the lower right corner is used to indicate a digital output (small black triangle is displayed on pins, configured as digital outputs). The color of the square resembles the current state of the pin – green for the activated (HIGH state) and white for the unactivated (LOW state).

To change the digital output state, first enable 'Enable output control' option, then either left or right click with mouse on the square representing the digital output to activate or deactivate this output.

Applicable PoKeys library commands

Read or set single pin state	
GetInput SetOutput	Reads current state of digital input Sets digital output state
Read or set single pin function	
GetPinData SetPinData	Reads pin function Set pin function
Read or set single pin key or macro mapping	
GetPinKeyMapping SetPinKeyMapping	Reads pin key mapping Sets pin key mapping options
GetPinMacroMapping SetPinMacroMapping	Reads pin macro mapping Set pin macro mappping options
Read or set triggered input mapping	
GetTriggeredInputMapping SetTriggeredInputMapping	Reads triggered input mapping options Sets triggered input mapping options
Read or set type-matic like delays and rates	
GetTypeMaticDelaysAndRates SetTypeMaticDelaysAndRates	Gets the typematic delay and repeat rate Sets the typematic delay and repeat rate

Read or write multiple inputs and outputs	
BlockGetInput1 / BlockGetInput2 / BlockGetInputAll55 BlockSetOutput1/BlockSetOutput2/ BlockSetOutputAll55	Read block of inputs (1 to 32) / (33 to 55) / (1 to 55) Set block of outputs (1 to 32) / (33 to 55) / (1 to 55)
Read or set multiple pin configurations	
GetAllPinConfiguration GetAllPinKeyCodes GetAllPinKeyMapping GetAllPinKeyModifierss SetAllPinConfiguration SetAllPinKeyCodes SetAllPinKeyMapping SetAllPinKeyModifierss	Read pin configuration for all pins Read pin key codes for all pins Read pin key mapping for all pins Read pin key modifiers for all pins Set pin configuration for all pins Set pin key codes for all pins Set pin key mapping for all pins Set pin key modifiers for all pins
Read or set automatic output initalization option	
GetAutoSetOutputs SetAutoSetOutputs	Get automatic outputs initialization option Set automatic outputs initialization option
Macro-related commands	
GetMacroKeys GetMacroName MacroCreate MacroDelete MacroGetActiveMacros MacroGetFreeSpace MacroGetLength MacroModifyLength MacroSaveConfiguration SetMacroKeys SetMacroName	Read key mapping for a macro Read macro name Reserve the space for a new macro Delete a specific macro Read the macro activation status Get free space for macros Read macro length Modify the length of a macro Save configuration of macros Set key mapping for a macro Set macro name
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetDigitalPinValues - SetDigitalPinValues - GetPinConfiguration - SetPinConfiguration	 Read digital inputs Set digital outputs Read pins configuration Set pins configuration
Raw basic device report	
GetFullDeviceReport	Set and read full device report

8.2. Connection signal output

Connection signal pin status can be set for pins 48 to 55. When USB connection with PC is established (or ethernet link is established on PoKeys56E), pin for which 'Connection signal pin status' is enabled, will be put into high state (or low state if pin polarity is setup as inverted). If the connection with PC

is lost and power through USB is still available (or ethernet link is lost on PoKeys56E), pin will go into low state (or high state if pin polarity is setup as inverted).

This command must not be confused with Failsafe settings, which are based on the communication timeout and not only link status.

PoKeys configuration software usage

To enable connection signal output, select one of the pins 48 to 55, then first select 'Digital output' option for this pin and check the 'Connection signal' option.

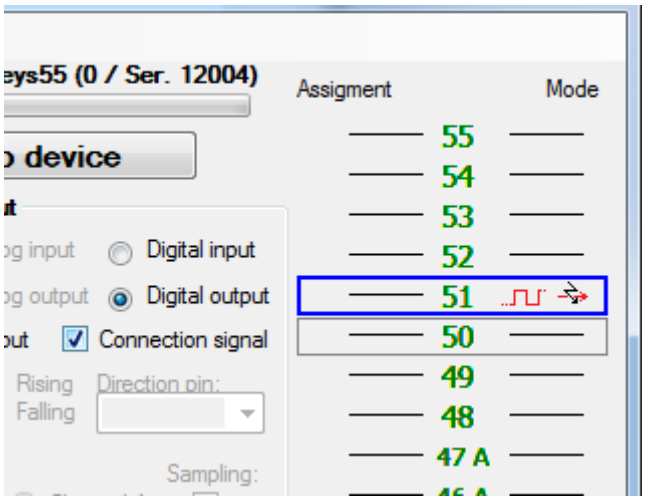


Figure 5: Connection signal setup

Applicable PoKeys library commands

Read or set connection signal setup	
GetConnectionSignalData SetConnectionSignalData	Retrieves connection signal pins data Sets connection signal pins data

8.3. Digital counters (only on PoKeys56 devices)

Selected pins of the PoKeys56U and PoKeys56E can be setup to count the number of signal transitions on those pins. Pin digital counter can be setup to be incremented/decremented on rising, falling or rising and falling edges of the input signal. If needed, additional pin can be selected to toggle between incrementing and decrementing mode.

For the list of pins that support digital counter option go to *Device pin functions* chapter of this manual.

If switches are used in the combination with digital counters, external debouncing circuit must be installed.

PoKeys configuration software usage

To setup digital counter inputs, first set the selected pin as 'Digital input' and if digital counter is available on the selected pin, the 'Enable counter' option will be enabled. Check this option and check 'rising' and/or 'falling' edge counting option.

To enable selection between incrementing or decrementing counter modes, a direction pin can be selected in the 'Direction pin' drop-down menu. If no pin is selected, the counter mode defaults to incrementing mode.

Digital counters values status page (menu 'Peripherals > Digital counters values…') can be used to check the proper working of the configured digital counters.

Applicable PoKeys library commands

Read or set single pin digital counter configuration	
GetPinData SetPinData GetDigitalCounterDirectionPin SetDigitalCounterDirectionPin	Reads pin data with counter options Set pin data with counter options Reads direction pins for the digital counters Sets direction pins for the digital counters
IsCounterAvailable	Checks if specified pin supports digital counter
Read or set multiple pin digital counter configuration	
GetAllPinConfiguration SetAllPinConfiguration	Read pin configuration for all pins with digital counter options Set pin configuration for all pins with digital counter options
Read or set digital counters values	
GetDigitalCountersValues	Reads current values of digital counters
ResetDigitalCounters	Resets values of digital counters
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetDigitalCounterValues - GetPinConfiguration - SetPinConfiguration	

8.4. Encoders

PoKeys devices can handle decoding of up to 26 (25 on PoKeys55) pairs of quadrature encoder signals. A and B signals of 25 'normal' encoders can be connected to any digital input and are intended for hand-driven rotational encoder switches with the quadrature signal frequencies up to 1 kHz.

Three fast encoders input pairs are available only on selected input pins (pins 1-2 as encoder 1, pins 3-4 as encoder 2, pins 15-16 as encoder 3 on PoKeys55 devices and pins 1-2 as encoder 1, pins 5-6 as encoder 2, pins 15-16 as encoder 3 on PoKeys56U and PoKeys56E devices) and can handle quadrature signal frequencies to about 100 kHz. When activated, fast encoders logically replace the 'normal' encoders 1, 2 and 3.

On PoKeys56U and PoKeys56E devices, ultra-fast encoder support is available on pins 8, 12 with the optional index signal input on pin 13. This can handle even higher frequencies.

Samilarly as simple digital inputs, encoders can be assigned to direct key mapping or keyboard macro (only with USB devices). This is possible for both directions (CW and CCW) separately.

If needed, encoder inputs can be incremented or decremented on every detected signal edge, increasing the resolution of the encoder for a factor of 4.

PoKeys configuration software usage

To enable encoder input on the selected pin, define the pin as digital input, select encoder index with numerical up-down selector and select appropriate encoder channel. The last step is to check the box ‘Encoder’.

If needed, encoder inputs can be incremented or decremented 4x faster, therefore each complete step will produce increment or decrement of 4 sub-steps. Using this setting, higher precision can be obtained.

To assign a key combination (only on USB devices) associated with the encoder, use the same procedure as described in the ‘Digital inputs and ouputs’ section. Each encoder is connected to PoKeys device with two signals and each of the pins that the encoder is connected to changes the keyboard mapping operation when encoder option is activated. Instead of being activated on each A and B signal front, keys are triggered on encoder value increment or decrement event. The key mapping settings however are setup on the pins that the encoder is connected to.

Enabling fast encoders

To enable fast encoders, go to menu 'Peripherals' and select 'Enable fast encoders'. Fast encoders inputs are fixed to pins 1,2 for fast encoder input 1, pins 3,4 for fast encoder input 2 and pins 15,16 for fast encoder input 3.

Displaying encoder raw values

To open encoder raw values dialog, go to Peripherals menu and select ‘Encoder RAW values’. The following dialog below appears. It simply shows the list of all encoders and their current values. In additional column, current encoder speed is displayed.

Status of the fast encoders is displayed in green, while the status of an ordinary encoder is displayed in light grey. Inactive encoders are displayed as dark grey boxes.

At the bottom of the window, there is a command button that can be used to reset the encoders' values.

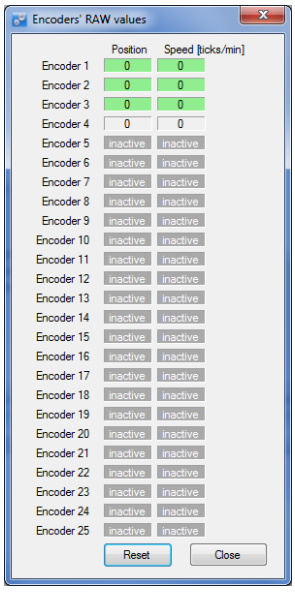


Figure 6: Encoders' RAW values

Applicable PoKeys library commands

Read single encoder value	
GetEncoderValue GetEncoderRAWValue	Read current encoder value Read encoder RAW value
Read or set single encoder configuration	
GetEncoderSettings SetEncoderSettings	Read encoder settings Set encoder settings
Read or set single encoder keyboard mapping configuration	
GetEncoderKeyMappingDirA GetEncoderKeyMappingDirB SetEncoderKeyMappingDirA SetEncoderKeyMappingDirB	Reads encoder mapping configuration for direction A Reads encoder mapping configuration for direction B Set encoder key mapping for direction A Set encoder key mapping for direction B
Read or set multiple encoders' values	
GetEncoders1_13_RAWValue GetEncoders14_25_RAWValue GetEncoders14_26_RAWValue	Read RAW value for encoders 1 to 13 Read RAW value for encoders 14 to 25 Read RAW value for encoders 14 to 26 (with ultra fast encoder as 26)
SetEncoders1_13_RAWValue SetEncoders14_25_RAWValue	Write RAW value for encoders 1 to 13 Write RAW value for encoders 14 to 25
ResetEncoderValue	Reset encoder value to 0
Read or set multiple encoders' configurations	

GetAllEncoderChannelPinMappings GetAllEncoderKeyMappingsDirA GetAllEncoderKeyMappingsDirB GetAllEncoderOptions	Get encoder pin mapping for all encoders Get encoder pin codes and modifiers for all encoders for direction A Get encoder pin codes and modifiers for all encoders for direction B Read encoder options for all encoders
SetAllEncoderOptions SetAllEncoderChannelPinMappings SetAllEncoderKeyMappingsDirA SetAllEncoderKeyMappingsDirB	Set encoder options for all encoders Set encoder pin mapping for all encoders Set encoder pin codes and modifiers for all encoders for direction A Get encoder pin codes and modifiers for all encoders for direction B
Read or set fast encoders configuration	
GetFastEncodersStatus GetFastEncodersStatusWOptions SetFastEncodersStatus SetFastEncodersStatusWOptions	Read fast encoder settings Read fast encoder settings and options Set fast encoder settings Set fast encoder settings
Read or set ultra-fast encoders configuration	
GetUltraFastEncodersSettings SetUltraFastEncodersSettings	Get ultra fast encoder settings Set ultra fast encoder settings
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetEncoderValues - SetEncoderValues - GetPeripheralConfiguration - SetPeripheralConfiguration	

8.5. Matrix keyboard

Matrix keyboard is a set of buttons, connected into a mesh. All buttons in a row share one contact, same goes for each of the buttons in the column. If a button is pressed, a key press is detected with a periodic scanning of each of the rows and columns. PoKeys devices use digital outputs for setting the voltage levels on rows and read column voltage levels using digital inputs that already have internal pull-up resistors, so no external circuitry is needed.

PoKeys devices support matrix keyboards of up to 16x8 in size, simpler 3x3, 4x3, 4x4 and others are of course fully supported.

Similarly to simple digital inputs, keys of the matrix keyboard connected to the PoKeys55/PoKeys56U device can be configured as USB keyboard keys. Direct mapping, mapping to macro sequence and triggered mapping are all supported. Additional alternate function can be used to assign two different keyboard keys to each of the matrix keyboard buttons. If additional (and freely selectable from the list of digital inputs) Fn+ input pin is inactive, the default function key is used. If the Fn+ key input pin is activated, an alternate function key is used instead of the default.

On all devices, the status of key presses of the matrix keyboard can be read using the PoKeys library commands without the need to setup the mapping described above.

Figure 7: Standard 4x3 matrix keyboard

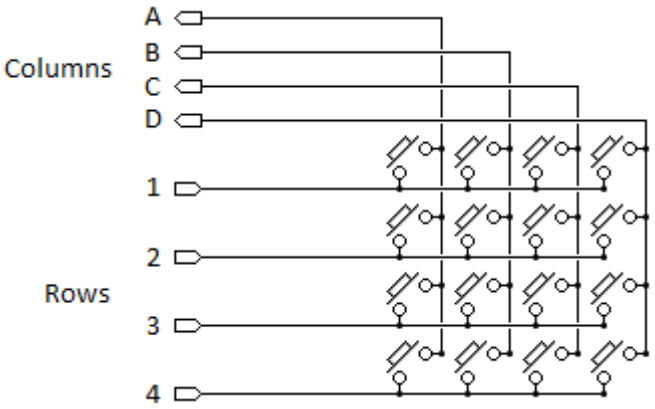


Figure 8: 4x4 matrix keyboard internal structure

PoKeys configuration software usage

Before any matrix keyboard configuration can be done, go to 'Peripherals > Matrix keyboard…', check the 'Enable matrix keyboard' option and select the number of rows and columns. Close the dialog and continue by selecting column and row pins.

Figure 9: Assigning row and column pins

Matrix keyboard column selection

Each free digital input pin can be assigned as matrix keyboard column input. Make sure the selected pin is configured as digital input, then check the 'Matrix keyboard' option for the pin and select the appropriate column letter from the list.

Matrix keyboard row selection

Each free digital output pin can be assigned as matrix keyboard row output. Make sure the selected pin is configured as digital output, then check the 'Matrix keyboard' option for the pin and select the appropriate row number from the list.

Keyboard mapping configuration

For configuring the keyboard mapping, open the 'Peripherals > Matrix keyboard…' menu. Matrix keyboard is schematically drawn in the dialog below. On the right, key mapping settings can be selected (on PoKeys55/PoKeys56U devices). To setup mapping, click on one of the keys in the matrix keyboard drawing and select appropriate key mapping options on the right.

To test the matrix keyboard, first make sure that the settings have been saved to device (close the Matrix keyboard dialog and click on 'Send to device' button). The matrix keyboard dialog can then be used to test the matrix keyboard – just press any key on your matrix keyboard and the appropriate button in the matrix keyboard drawing will be highlighted.

To setup different key presses for 'key press' and 'key release' events, check 'Triggered mapping' option and select different settings for 'Down key' ('key press' event) and 'Up key' ('key release' event).

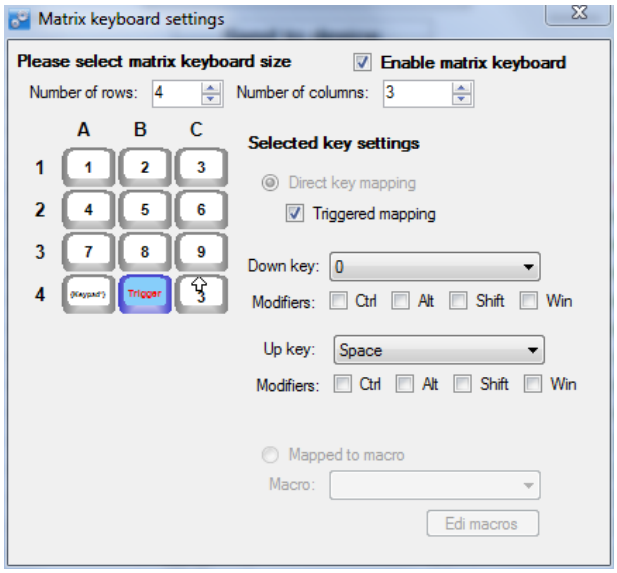


Figure 10: Matrix keyboard configuration for a 4x3 matrix keyboard

Applicable PoKeys library commands

Read matrix keyboard status	
GetMatrixKeyboardKeyStatus	Read matrix keyboard status
Read or set basic matrix keyboard configuration	
GetMatrixKeyboardConfiguration SetMatrixKeyboardConfiguration	Read matrix keyboard configuration Set matrix keyboard configuration
Read or set matrix configuration with alternate function support	
GetMatrixKeyboardConfiguration_AlternateFunctionSupport SetMatrixKeyboardConfiguration_AlternateFunctionSupport	Read matrix keyboard configuration (for alternate functions mapping) Set matrix keyboard configuration (for alternate functions mapping)
Read or set matrix configuration with triggered mapping support	
GetMatrixKeyboardConfiguration_TriggerSupport SetMatrixKeyboardConfiguration_TriggerSupport	Read matrix keyboard configuration (for triggered mapping) Set matrix keyboard configuration (for triggered mapping)
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetMatrixKeyboardConfiguration - SetMatrixKeyboardConfiguration - GetMatrixKeyboardKeysStatus	

8.6. Analog inputs

Analog input function is only available for pins 43 to 47 (on PoKeys55 devices) or for pins 41 to 47 (on PoKeys56 devices). On PoKeys55 and PoKeys56U devices, these analog inputs can also be freely mapped to any of the 6 joystick axis; X, Y, Z, rotation X, rotation Y and throttle.

Analog inputs (10-bit on PoKeys55 and 12-bit pn PoKeys56U/PoKeys56E) are sampled at a fixed rate of 10 kHz and fed through adjustable discrete low-pass filter with the following equation

$$y(k) = y(k-1) * \frac{filter}{filter+1} + u(k) * \frac{1}{filter+1}$$

where y(k) is the output analog value, u(k) is a new A/D sample and *filter* is a user-adjustable constant. Sample u(k) is produced according to the following equation

PoKeys56U/PoKeys56E	PoKeys55
$\boldsymbol{u(k) = \frac{U(k)[V]}{3.3\ V} * 4095}$	$u(k) = \frac{U(k)[V]}{3.3\ V} * 1023$

where U(k) (in Volts) is a voltage present on the selected analog input pin.

PoKeys configuration software usage

To open analog inputs dialog, go to Peripherals menu and select 'Analog inputs and outputs'. Dialog below appears. To enable display of analog input channel, check the appropriate check box. It is enabled only when the input is set up as analog input.

The progress bar displays the current voltage at the pin with the maximum at 3.3V. Below the input selection boxes user can set low-pass filtering for analog inputs. When analog input signal appears to be flickering or jumping due to analog signal noise, move the value for the filter to the right towards label 'slow signals' and then press Set button.

If pin 43 is set as analog output (only on PoKeys55 devices), analog value can be set for this pin.

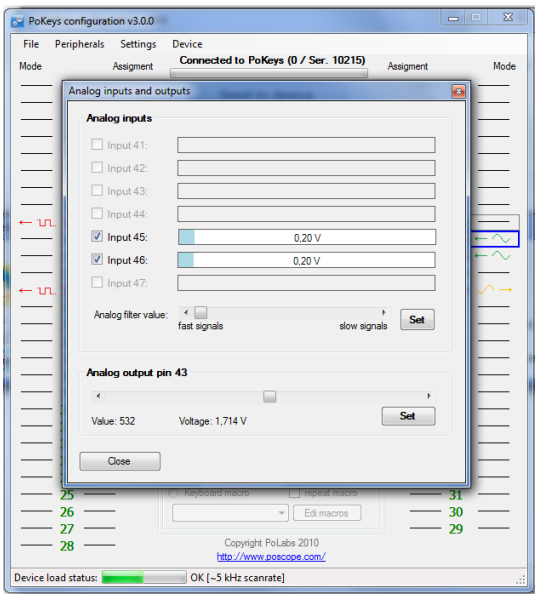


Figure 11: Analog inputs and outputs dialog

Applicable PoKeys library commands

Read single analog input value	
GetAnalogInput	Read analog input value
Read or set single pin function	
GetPinData SetPinData	Reads pin function Set pin function
Read all analog inputs	
GetAllAnalogInputs	Read all analog inputs in one command
Read analog input maximum value	
GetAnalogValueMax	Returns the maximum value of analog input
Read or set analog low-pass filter parameter	
GetAnalogRCFilterValue SetAnalogRCFilterValue	Read low-pass filter parameter Set low-pass filter parameter
COM_Execute command	
COM_ExecuteInner with the following parameters: - GetAnalogPinValues	

8.7. Joystick mapping

Each axis of the PoKeys virtual joystick can be assigned an analog input source. In addition, analog to digital mapping option can be enabled, which allows user to connect an analog joystick to a PoKeys devices and simulate key presses for each direction of the joystick. User can freely select dead band and saturation ranges.

Besides mapping the analog inputs to virtual joystick axes, digital inputs (or encoder switching events) can be mapped to any of the 32 virtual joystick buttons (either directly or 'triggered') or 4-way POV hat selector. The triggered mapping to joystick buttons enables used to select different pins that triggers selected joystick button on off-to-on transition (Down Event) and on on-to-off transition (Up Event).

PoKeys configuration software usage

Joystick axis and buttons mapping can be setup via Joystick mapping dialog. Go to 'Peripherals' and select 'Joystick settings…'. The dialog on Figure 12 appears.

Each axis can be assigned an analog input. In addition, analog to digital mapping option can be enabled. This allows user to connect an analog joystick to a PoKeys devices and simulate key presses for each direction of the joystick. To do so, first check 'Map to key' option. Then set the dead band (when input value will be between lower and upper dead band margins, no keys will be activated) using sliders. In the lower part of the window, select the mapping options.

For simple direct mapping (pin input status is directly reflected in joystick button status) use the 'Direct mapping' option and select pin number to be associated with selected joystick button. If more advanced behavior is needed (joystick button is pressed for a short time only on transitions of pin status), user should select 'Triggered mapping' option to select one pin that triggers selected joystick button on off-to-on transition (Down Event) and one pin that triggers this joystick button on on-to-off transition (Up Event).

If joystick button mapping is to be used in connection with encoder inputs, use 'Triggered mapping' option (encoder's values cannot be directly translated into direct mapping) and select a pin with the appropriate encoder channel. For example: pins 5 and 6 are set up as digital inputs with encoder (channel A on pin 5 and channel B on pin 6). When pin 5 is selected as Down event pin for joystick Button 2 and pin 6 is selected as Down event pin for joystick Button 3, rotating the encoder in positive direction will trigger joystick Button 2 on each detent. Similarly, rotating the encoder in negative direction, joystick Button 3 will be triggered on each detent. Up Event pin option cannot be used in connection with encoders.

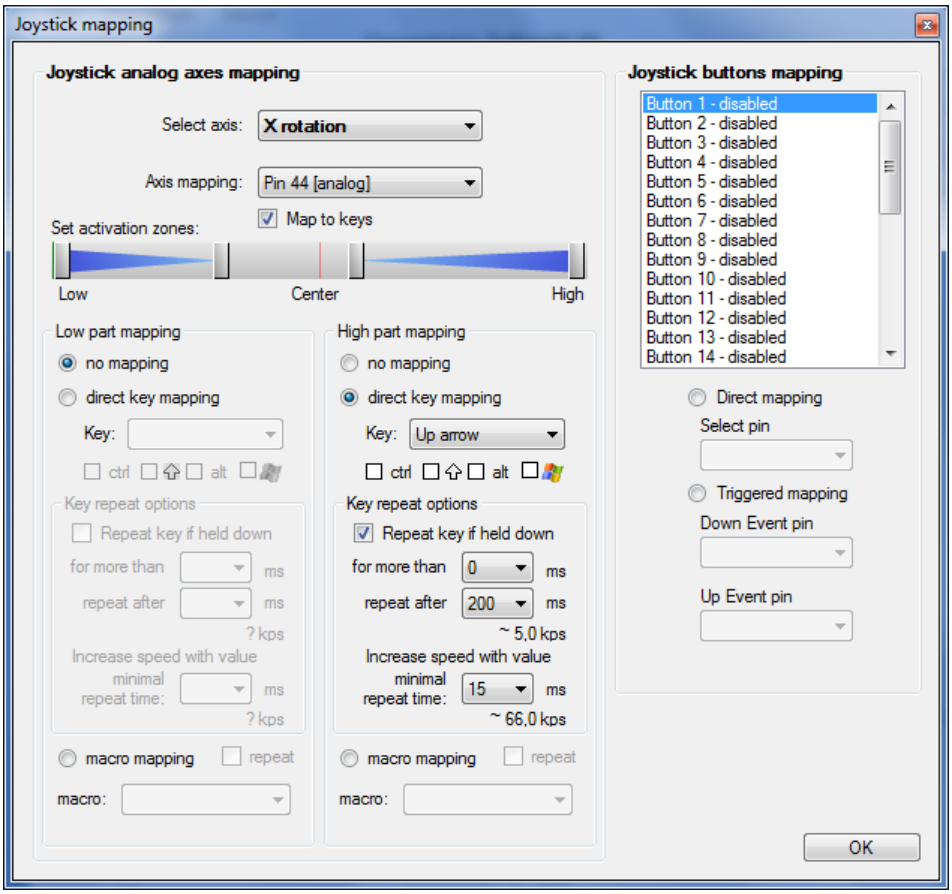


Figure 12: Joystick mapping settings

Applicable PoKeys library commands

Read or set joystick axes and buttons mapping	
GetJoystickMapping SetJoystickMapping	Read joystick mapping options Set joystick mapping options
Read or set joystick triggered mapping	
GetJoystickTriggeredMapping SetJoystickTriggeredMapping	Read joystick triggered mapping options Set joystick triggered mapping options
Read or set analog to digital mapping	
GetJoystickDigitalMapping SetJoystickDigitalMapping	Read joystick analog to digital mapping configuration Set joystick analog to digital mapping configuration

8.8. PWM outputs

PoKeys55, PoKeys56U and PoKeys56E devices support PWM output function on 6 pins (pins 17 to 22). Different duty cycles can be assigned to each PWM output however, all outputs share the same PWM period. PWM outputs can be easily amplified using an external transistor and used for control of loads with increased current demand. PoKeys PWM outputs can also be used to drive various R/C servo motors that accept PWM signal with 50 Hz frequency (20 ms PWM period) and duty cycles between 5 and 10 % (1 to 2 ms).

PoKeys devices have an in-built PWM module that operates at a fixed clock frequency (12 MHz in PoKeys55 devices and 25 MHz in PoKeys56U and PoKeys56E devices). Both the PWM period and the PWM duty cycles must be expressed as number of module clock cycles (i.e. 20 ms PWM period equates to 0.020 x 25 000 000 = 500 000 on PoKeys56U/E devices).

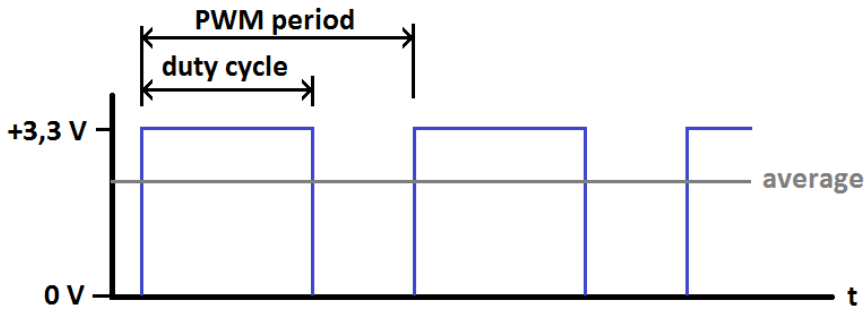


Figure 13: PWM output

PoKeys configuration software usage

PoKeys device's PWM (pulse width modulation) module can be setup via Peripherals > PWM outputs….

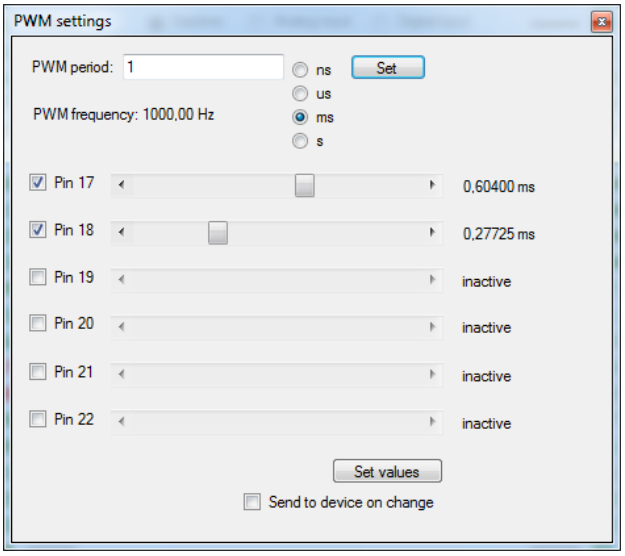


Figure 14: PWM outputs settings

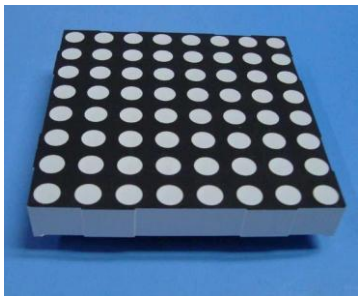
In this window, user can enter PWM period and set PWM duties for each channel. Channels can be independently enabled or disabled. After a change is made, user must click 'Set values' button or

check 'Send to device on change' checkbox. Left position of a slider means 0% and right position 100% respectively.

Applicable PoKeys library commands

Read or set PWM outputs configuration	
SetPWMOutputs GetPWMOutputs	Set PWM period and duty cycles Get PWM period and duty cycles
Read PWM module frequency	
GetPWMFrequency	Read the PWM module base frequency
COM_Execute command	
COM_ExecuteInner with the following parameters: - SetPWMOutputs	

8.9. Matrix LED output



PoKeys devices support 2 additional external 8x8 LED matrix displays. Each display is based on 2 8-bit shift registers, one controlling the row and the other controlling the column signals. PoKeys refreshes the display at the rate of 2800 rows per second (which gives around 350 Hz refresh rate of the whole 8x8 display).

Once enabled, PoKeys device assigns pins 9, 10 and 11 to be used with display 1, and pins 23, 24, 25 to be used with display 2.

Function	Matrix LED display 1	Matrix LED display 2
data	9	23
latch	10	24
clock	11	25

Table 1: Matrix LED displays pin assignments

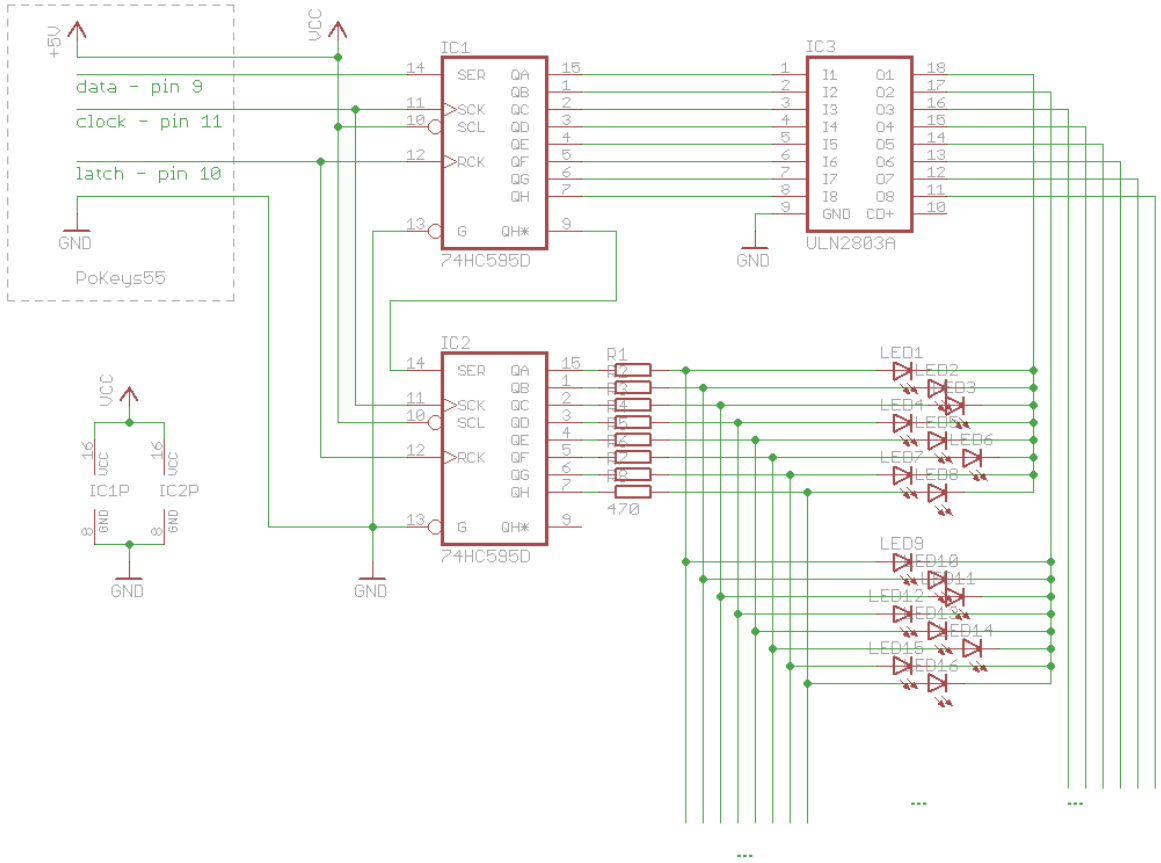


Figure 15: Matrix LED display deserialization circuit

PoKeys configuration software usage

To set-up matrix LED displays, open 'Peripherals > LED displays...'. The following dialog appears

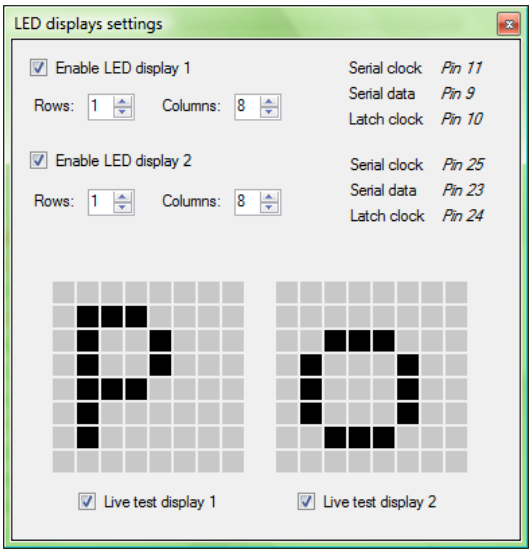


Figure 16: Matrix LED setup dialog

For each of the displays, number of rows and columns can be selected. On the bottom, after enabling 'Live test display x', user can test the displays in real-time by clicking on gray rectangles. Left-click turn-s selected pixel on, while right-click resets selected pixel.

Applicable PoKeys library commands

Read or set matrix LED settings	
MatrixLEDGetSettings MatrixLEDSetSettings	Get matrix LED settings Set matrix LED settings
Matrix LED commands	
MatrixLED1ClearAll MatrixLED1SetPixel MatrixLED1Update MatrixLED2ClearAll MatrixLED2SetPixel MatrixLED2Update	Clear matrix LED 1 Set pixel on matrix LED 1 Matrix LED 1 update Clear matrix LED 2 Set pixel on matrix LED 2 Matrix LED 2 update
COM_Execute command	
COM_ExecuteInner with the following parameters: - SetMatrixLEDconf - SetMatrixLEDstat	

8.10. LCD

PoKeys devices support connecting one alphanumeric LCD module up to a size of 4x20 (4 rows, 20 columns). The selection of the module is limited by support for HD44780 or compatible chipset. Usually these displays come in various sizes - 1/2/4 line with 8/16/20 characters and colors (black letters on green background, white letters on blue background ...).

Figure 17: Typical 2x16 character LCD

These displays share standard pin-out that is listed in the table below:

Pin	Symbol	Function	PoKeys pin
1	**Vss**	Ground	**GND**
2	**Vdd**	Positive supply (usually 5V)[2]	**5V (usually) or 3.3V**
3	**Vo**	Contrast adjustment	**Variable resistor between GND and supply or PWM output**
4	**RS**	Instruction/data input	**Pin 29**
5	**R/W**	Read/write	**Pin 28**
6	**E**	Enable signal	**Pin 30**
7	DB0	Data bus – bit 0	Not connected
8	DB1	Data bus – bit 1	Not connected
9	DB2	Data bus – bit 2	Not connected
10	DB3	Data bus – bit 3	Not connected
11	**DB4**	Data bus – bit 4	**Pin 26 (or secondary 34)**
12	**DB5**	Data bus – bit 5	**Pin 25 (or secondary 33)**
13	**DB6**	Data bus – bit 6	**Pin 24 (or secondary 32)**
14	**DB7**	Data bus – bit 7	**Pin 23 (or secondary 31)**
15	Backlight (optional)		
16	Backlight (optional)		

Table 2: LCD pin assignments

LCD display can be used to display various data. A third-party application or a script can execute all supported operations, including LCD initializing, clearing, moving cursor, setting display shifting mode, custom character defining and displaying text.

[2] Positive supply voltage depends on LCD used. User should find this information in datasheet of the LCD in use. **On PoKeys56E take special care that the power supply voltage for the LCD is adequate (there is 0.8 V of voltage drop between power supply input and 5V pin).**

PoKeys56U and PoKeys56E devices by default function in buffered LCD mode. In this mode, any LCD-related command that is sent to PoKeys device is first buffered and when possible, PoKeys device executes the LCD refresh on its own. In this mode, some LCD operations are not operational – cursor movement is controlled by the PoKeys device and cursor move or display commands may not work as expected. If this low-level control is desirable, buffered mode must be deactivated first. In other cases, it is advisable to use buffered mode in order to allow better load balancing in PoKeys devices and in the end attain greater communication speed.

Before LCD initialization, the LCD module size (number of rows and columns) must be specified and appropriate pin assignment (primary or secondary, see the table above) must be selected. Secondary pins must be selected in case of matrix LED display 2 in use.

PoKeys configuration software usage

Functions of this interface can be tested through PoKeys settings application. Just open Peripherals > Test LCD… and dialog below will appear.

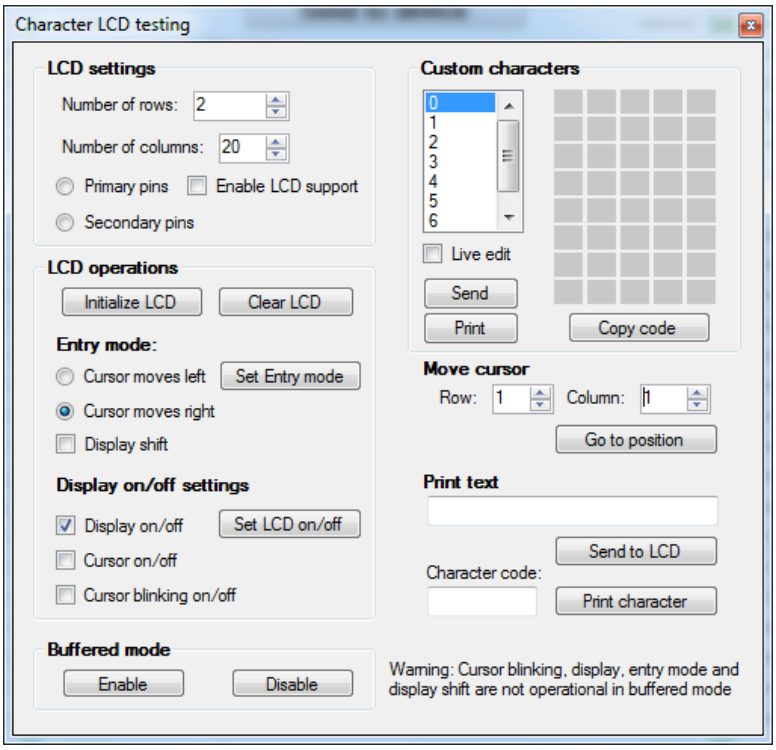


Figure 18: Character LCD testing dialog

LCD settings

In this part, user can set number of rows and columns in the LCD used. Support for LCD can be enabled or disabled also. Data pins for LCD can be selected on primary (23 to 26) or secondary (31 to 34) pins. ***Secondary pins must be selected in case of matrix LED display 2 is in use.***

LCD operations

Before user can start using the LCD, LCD module must be initialized. This is done via 'Initialize LCD' button. Button 'Clear LCD' clears LCD display and moves cursor to home position.

User can also set entry mode settings of LCD module. Cursor can be set-up to move either right (normally) or left after each character displayed. If 'Display' shift is enabled, whole display shifts with every new character displayed.

Settings are processed after user clicks button 'Set Entry mode' and work only in ‘unbuffered’ mode described above.

Display on/off settings

User can set on/off switches for whole display, cursor and cursor blinking.

Settings are processed after user clicks button 'Set LCD on/off' work only in ‘unbuffered’ mode described above.

Custom characters

Simple interface enables to draw up to 8 custom characters. These characters can then be used on display. Selecting 'Live edit' mode will transfer the character each time a change is made to any of the pixels. Character can be previewed via button 'Print', which puts current custom character on the LCD display.

Move cursor

This section enabled user to move cursor to any position on the screen works only in ‘unbuffered’ mode described above.

Print text

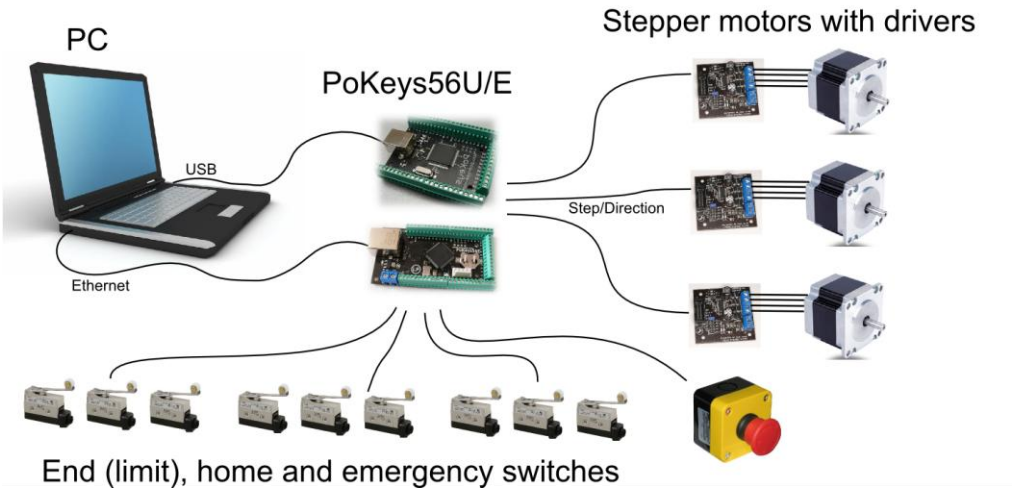
Sends entered text to display module. If advanced characters are needed, enter character code in lower text box and press 'Print character'.

Applicable PoKeys library commands

Read or set LCD module configuration	
LCDGetSettings **LCDSetSettings**	Read LCD module settings Set LCD module settings
LCD operations	
LCDInit **LCDClear** **LCDSetEntryMode** **LCDDisplayOnOffControl** **LCDSetMode** **LCDGotoXY**	Initialize LCD display Clear LCD display Set entry mode for LCD Set LCD display on/off status Set LCD display mode Move cursor to a specified position in the display
Display text on LCD	
LCDPrint **LCDPutc**	Print a string on the LCD display Put a single character on LCD display
LCDDefineCustomCharacter	Define a custom character in the LCD display memory
COM_Execute command	
COM_ExecuteInner with the following parameters: - LCD	

8.11. Pulse engine

PoKeys Pulse engine (available on PoKeys56U and PoKeys56E devices) enables the control of up to three stepper motors. PoKeys Pulse engine can work in standalone (rapid positioning and speed control mode, homing, jogging via kbd48CNC keyboard) or in slave mode (under the command of the PC application). For each stepper motor, two end (or limit) switches and one home switch are supported and can be configured by the software. PoKeys Pulse engine also supports a dedicated emergency switch input, which stops the pulse generation.



PoKeys device pins in use

Pin	Function
32	Limit+ switch – x
33	Limit+ switch – y
34	Limit+ switch – z
35	Limit- switch – x
36	Limit- switch – y
37	Limit- switch – z
38	Direction output – x
39	Direction output – y
40	Direction output – z
46 !	Step output – x
48 !	Step output – y **(external 470 Ω pull-up resistor required)**
49 !	Step output – z **(external 470 Ω pull-up resistor needed)**
47	Home switch – x
50	Home switch – y
51	Home switch – z
52	Emergency switch input
53	Safety charge pump 5 kHz output

Pulse engine operating principles

PoKeys Pulse engine divides the operation in 1 millisecond time slots, during which the pulse frequency is held constant, and supports the generation of up to 25 pulses per 1 millisecond time slot (which equates to 25 kHz maximum pulse frequency supported).

At each time slot beginning, the limit and home switches are read and evaluated. If emergency switch or any limit switch is activated (enabled in the configuration) and the switch is tripped, the pulse engine is put into STOP mode and no more pulses are generated (with hard-stop mechanism). However, when in standalone jogging mode, the limit switches are ignored. If STOP condition is not met, the Pulse engine uses the data in the slot buffer to set the pulse frequency in the next time step.

The slot buffer is 128-slots deep buffer that holds pulse frequencies for each axis, giving a 128 millisecond buffered motion period. Each slot entry in the buffer holds 3 bytes (1 byte per axis) and each axis entry uses 7-bits for pulse frequency and MSB bit for the direction signal (if MSB bit is set, the direction output is activated).

Fill buffer command is used to transfer the data to the slot buffer. Application that uses the fill buffer command should send as much time slot data as possible. PoKeys Pulse engine will then return the number of accepted time slots. This omits the need of additional query on the buffer free space in PoKeys Pulse engine buffer. Application should only then increase the 'read pointer' based on the number of accepted slots. Additionally, fill buffer command returns a number of parameters of the pulse engine (position, engine state, state of limit and home switches and states of each axis).

Modes of operation

- **Stopped**: the pulse engine does not generate any pulses.
- **Error**: the pulse engine encountered an error (e.g. limit switch was activated).
- **Homing**: homing mode is activated. In this mode, one or more axes can be homed. The selected axis (or axes) moves in negative direction at predefined fraction of the maximum speed until the home switch is tripped. Then, the direction is changed to positive and speed decreased to half the previous speed. When the switch is tripped, the internal position counter is reset and the axis is commanded to stop. This operation does not include moving back to position 0. The state of homing procedure is reflected in axes states.
- **Jogging**: each axis can be moved in jogging mode. Jogging mode also overrides the limit switches. Only set_speed internal controller is functional in this mode with speed limited by the axis parameters.
- **Buffer**: the internal controller is disabled and the slots are fed direclty from a slot buffer, which must be constantly filled by the external application.
- **Internal**: the internal controller supports two modes:
    - Internal position control: moves the axis to the desired position, following the limitations set by the axis parameters,
    - Internal speed control: moves the axis at the desired speed, following the limitations set by the axis parameters.

Both internal modes and buffer mode utilize the internal buffer for operation. Therefore the fluctuations in the free space of the slot buffer can be observed in this modes. Before utilizing the buffer mode after internal controller mode, external application must either wait 10 ms for the buffer to clear or clear the buffer using the clear buffer command.

Safety charge-pump output

When activated, 5 kHz square safety charge-pump signal is present on pin 53 if the Pulse engine is in normal operating mode (i.e. not in Stopped or Error mode).

Axis parameters

Internal mode uses the following axis parameters of motion:

- Maximum speed: maximum frequency of pulses (in pulses/s)
- Acceleration: maximum acceleration (in pulses/s^2)
- Deceleration: maximum deceleration (in pulses/s^2)
- Limit and home switches configuration: each of the 6 limit and 3 home switch inputs can be enabled or disabled
- Direction change configuration: direction can be changed separately for each of the axes
- Homing direction configuration: direction of homing can be changed separately for each of the axes

PoKeys configuration software usage

Functions of the PoKeys Pulse engine can be tested using the 'Pulse engine status/control' dialog, accessed by a separate menu item 'Pulse engine'.

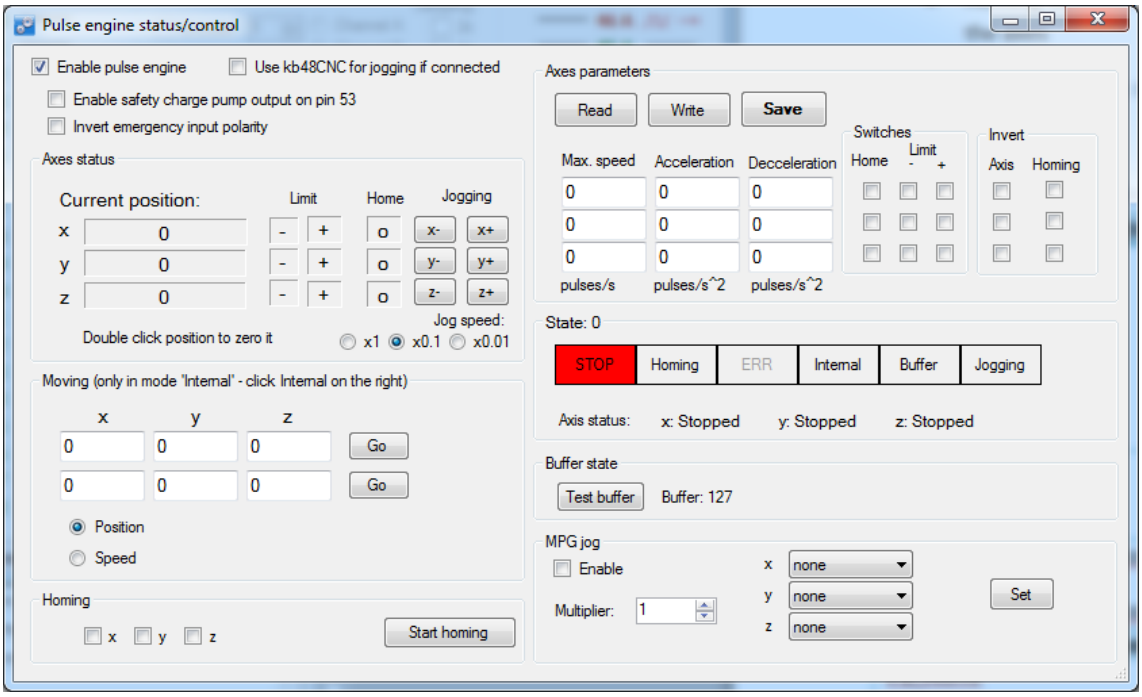


Figure 19: PoKeys Pulse engine status/control dialog

Applicable PoKeys library commands

Read or set pulse engine configuration and status	
PE_GetParameters	Read pulse engine parameters (maximal speed, acceleration, deceleration)
PE_SetParameters	Set pulse engine parameters (maximal speed, acceleration, deceleration)
PE_GetInfo	Read pulse engine information (number of supported axes, maximum pulse frequency, motion buffer depth, motion

	slot timing)
PE_GetStatus PE_SetStatus	Read the pulse engine status (current axes positions and states, switches status, main pulse engine state and status) Set pulse engine enabled state
PE_GetAxesConfiguration PE_SetAxesConfiguration	Read axis configuration (move direction, homing direction, presence of home and limit switches) Set the axis configuration
Internal controller commands	
PE_Move PE_SetCurrentPosition	Execute a move using the internal controller (pulse engine must be in internal controller mode for this command to work) Set current axes position register value
PE_StartHoming	Execute homing for the selected axes
Buffered mode commands	
PE_ClearBuffer PE_FillBuffer PE_GetFreeBufferSize	Clear (purge) the internal motion buffer Send motion buffer entries Read the motion buffer empty space

8.12. PoExtBus

PoExtBus bus support enables user to add additional 10 8-bit shift registers to the project based on PoKeys device. This gives additional 80 digital outputs that can be easily controlled with included dll interface or other 3rd party interface for PoKeys device.

On PoKeys55, device assigns pins 35, 36 and 37 to be used with PoExtBus. On PoKeys56 series, there is a dedicated connector on the board, which serves for the PoExtBus functions.

Function	ExtBus – PoKeys pin
Clock	35
Data	36
Latch	37

Table 3: PoExtBus pin assignments

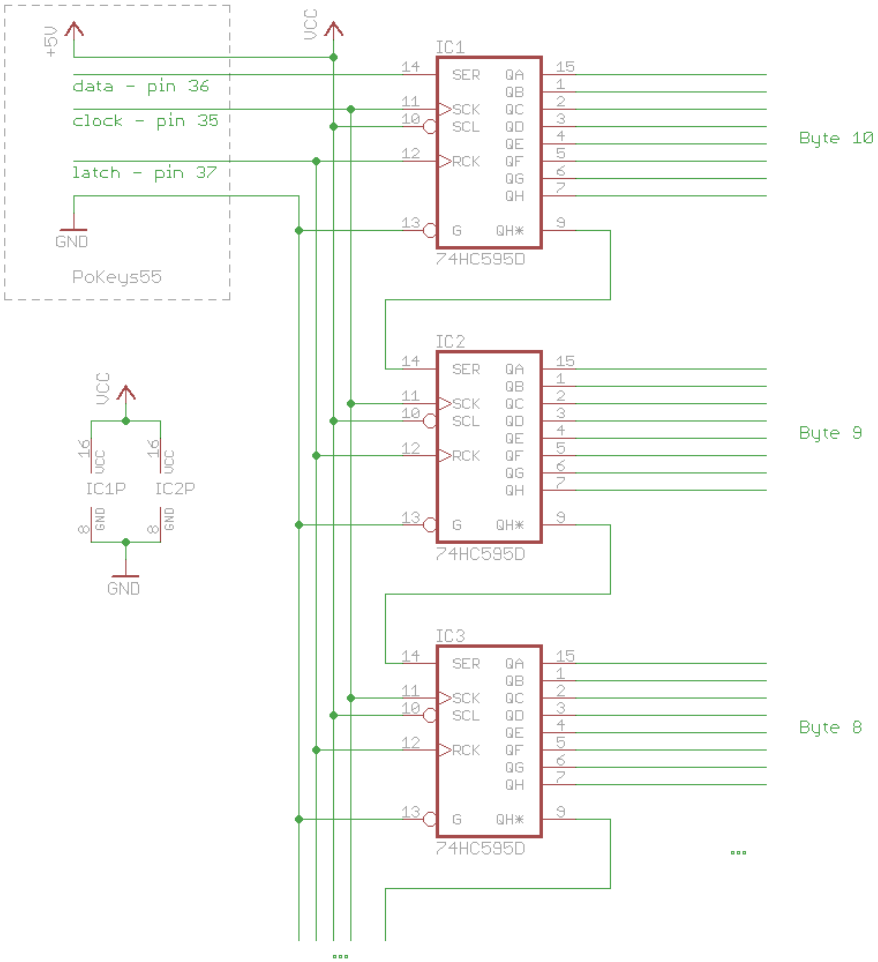


Figure 20: PoExtBus deserialization circuit

PoKeys configuration software usage

To set-up and test PoExtBus, open 'Peripherals' > PoExtBus...'. The dialog below (Figure 21) appears. By using mouse left and right clicks, user can turn on or off each of the outputs.

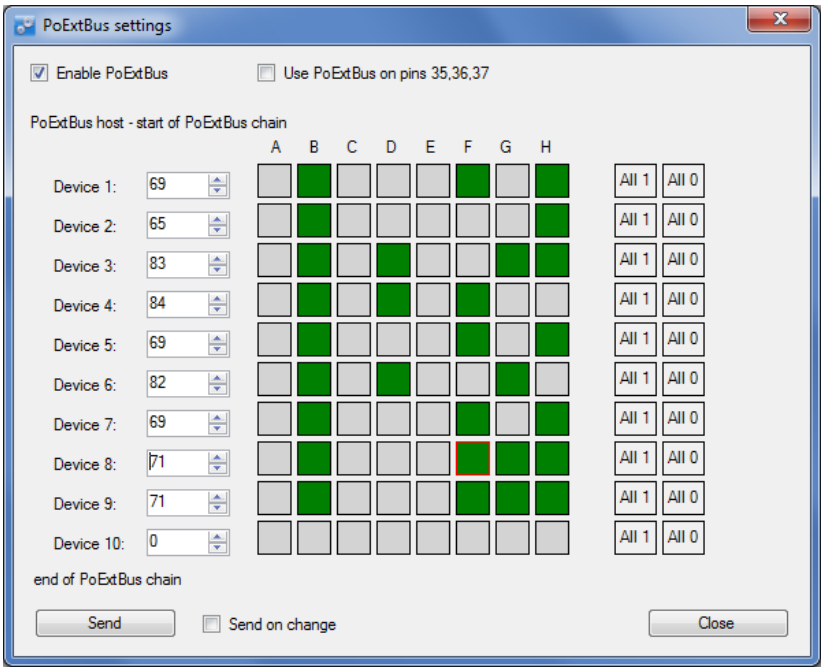


Figure 21: PoExtBus setup dialog

Applicable PoKeys library commands

Read or set PoExtBus configuration and status	
AuxilaryBusGetData	Read PoExtBus configuration
AuxilaryBusSetData	Set PoExtBus configuration and data

8.13. PoNET

PoKeys56 devices support PoNET devices that extend PoKeys56 functionality. Multiple PoNET and PoExtBus devices can be linked together. While the PoNET devices and PoExtBus devices share the same ExtBus connector, found on PoKeys56 boards, user should pay attention in connecting devices of both types together (see the schematics below).

All PoNET devices must be connected in parallel to each other and directly to the PoKeys56 board, while the PoExtBus devices should be connected in series after the PoNET devices.

Adding new devices

After connecting new PoNET device, go to 'Peripherals > PoNET...'. The following dialog will appear

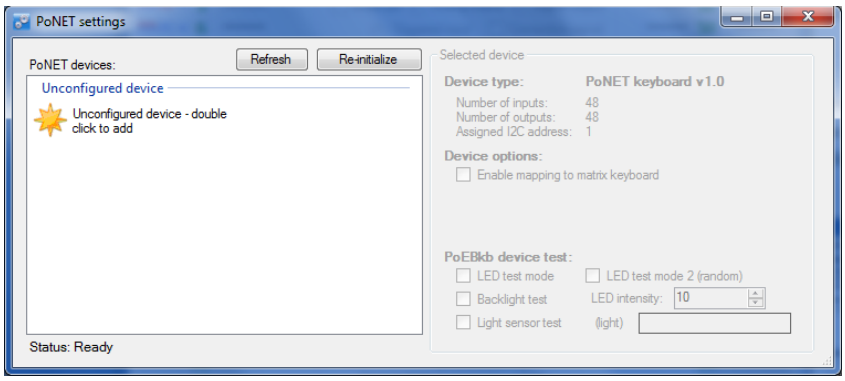


Figure 22: PoNET settings dialog with an unconfigured device

In order to register new device, double click on the 'Unconfigured device' icon. In the next 10 seconds press any key on the device that is about to be added. If the process is successful, status LED on the device will stop blinking and will be constantly lit. The device will also be listed as in the dialog below:

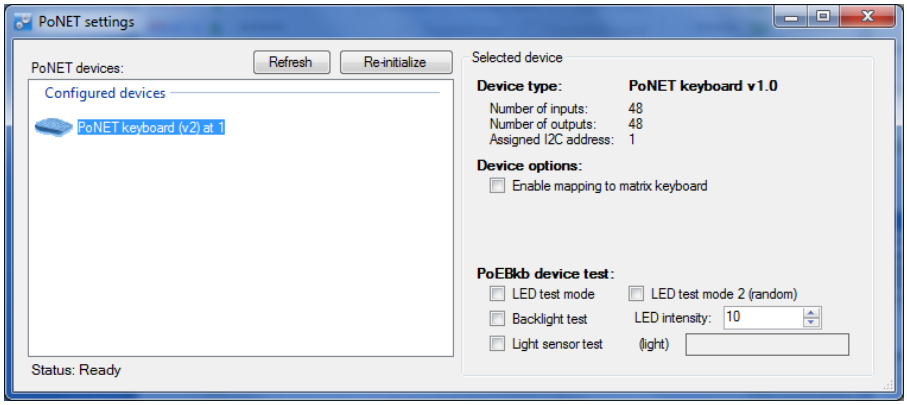


Figure 23: PoNET settings dialog with a successfully configured device

PoNET kb48CNC keyboard

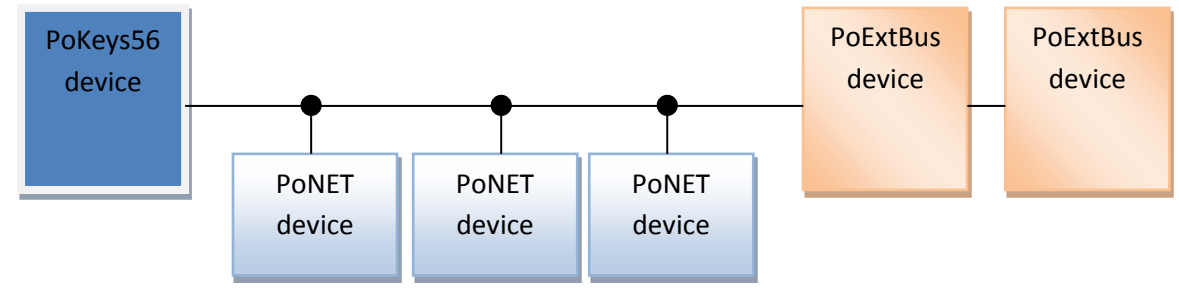
The device can be virually mapped to PoKeys matrix keyboard. Third party software can set the status of LEDs under the keys, read the light sensor that measure the amount of light in the environment and set the intensity of the LEDs.

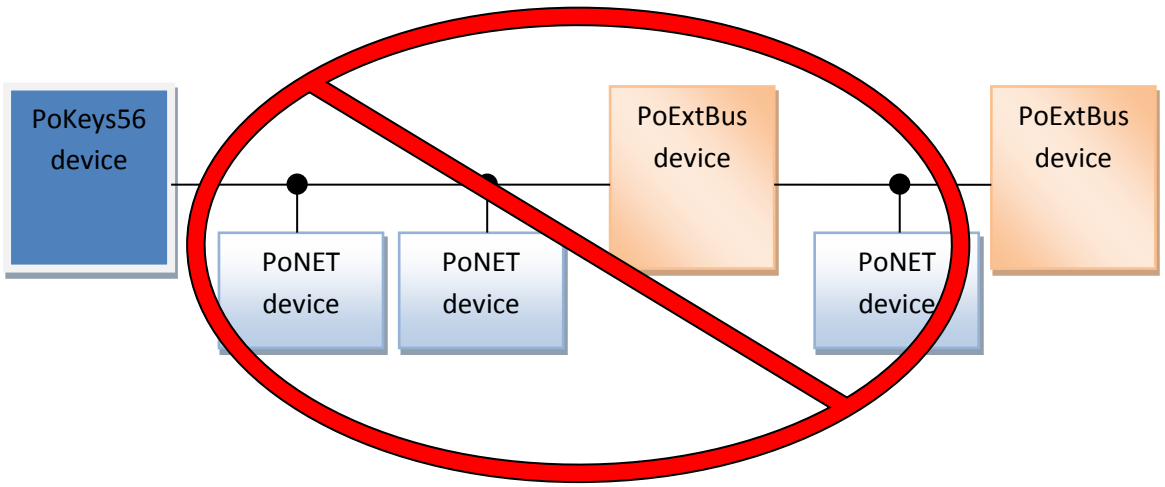
To map the PoNET keyboard to PoKeys matrix keyboard, initialize the PoNET bus as described above in 'Adding new devices', select the keyboard in the device list and check the checkbox 'Enable mapping to matrix keyboard'.

PoNET/PoExtBus connection example

Pay attention to the following:

- **PoNET device can be connected only to PoKeys56 board or other PoNET device**
- **PoExtBus device can be connected to either PoKeys board, PoNET or PoExtBus device.**
- **There should be not PoExtBus device connected between any PoNET devices and PoKeys board**





Applicable PoKeys library commands

Device discovery and initialization commands	
PoNET_AddDeviceStart	Starts 'Register device' procedure
PoNET_AddDeviceStatus	Retrieves the status of 'Register device' procedure
PoNET_AddDeviceStop	Stops the 'Register device' procedure
PoNET_CheckForNewDevices	Checks for new devices
PoNET_ReinitializePoNET_noClear	Unregisters all PoExtBus Pro devices (but keeps the configuration)
PoNET_ReinitializePoNET_wClear	Unregisters all PoExtBus Pro devices and clears the configuration
Read internal state of the PoI2C engine in PoKeys	
PoNET_GetState	Function returns the internal PoI2C state of the PoKeys device. Look in the protocol specification document under PoI2C settings and communication
Read or write to selected PoNET device	
PoNET_GetFWversion	Reads firmware version
PoNET_GetLightIntensity	Reads light intensity using light sensor
PoNET_GetModuleSettings	Retrieves module settings
PoNET_GetModuleStatus	Reads data from the module
PoNET_SetModuleSettings	Sets the module options
PoNET_SetModuleStatus	Writes data to module
PoNET_SetPWM	Sets PWM duty cycle for LED intensity

8.14. Failsafe settings

PoKeys56U and PoKeys56E devices support the configuration of the failsafe state for the digital outputs, PWM outputs, PoExtBus devices and PoKeys Pulse engine.

When the communication with the device is interrupted for longer than a period defined in the failsafe configuration, peripherals listed above enter the failsafe mode, which can be setup in 'Failsafe settings' dialog (Peripherals > Failsafe settings…)

Peripheral	Failsafe setting
Digital outputs	Active Off / Active On
PoExtBus outputs	Active Off / Active On
PWM outputs	Fixed duty cycle in %
Pulse engine	No setting – Pulse engine enters emergency mode on failsafe activation

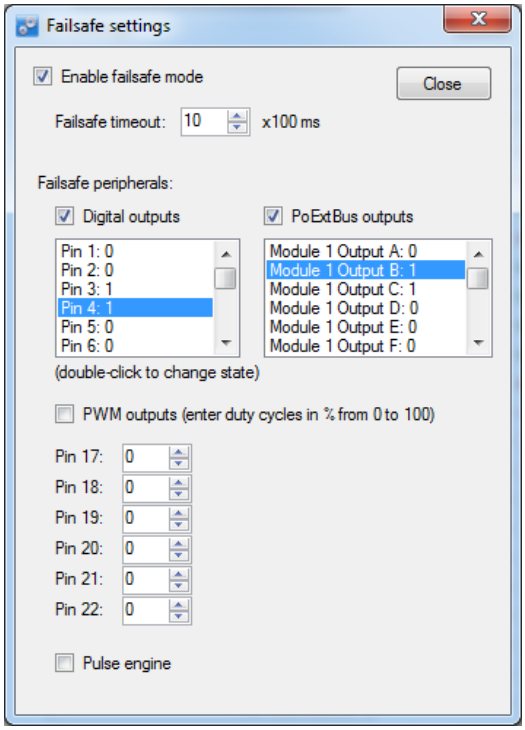


Figure 24: Failsafe settings dialog

Applicable PoKeys library commands

Failsafe configuration	
N/A	N/A

8.15. I²C protocol

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006. The name I2C translates into "Inter IC". Sometimes the bus is called IIC or I²C bus.

PoKeys56U and PoKeys56E devices support communication with I²C slave devices, connected to the PoExtBus connector. As I²C, PoNET and PoExtBus use the same connector, PoExtBus, PoNET and I²C functions are automatically switched by the PoKeys device.

However, in case of problems with noise caused by the I²C or PoNET devices communication, PoExtBus functionality can be moved to pins 35,36,37 as with PoKeys55 devices).

Marking the pin closer to the bottom of the board (the oposite side of either Ethernet or USB connector) as pin 1, the I²C devices should be connected as follows:

Pin 1	**Power supply 3.3V**
Pin 2	Ground
Pin 3	Serial data
Pin 4	
Pin 5	Serial clock

Protocol can be tested via PoKeys configuration software. Click on Peripherals > I2C bus test... The following dialog appears.

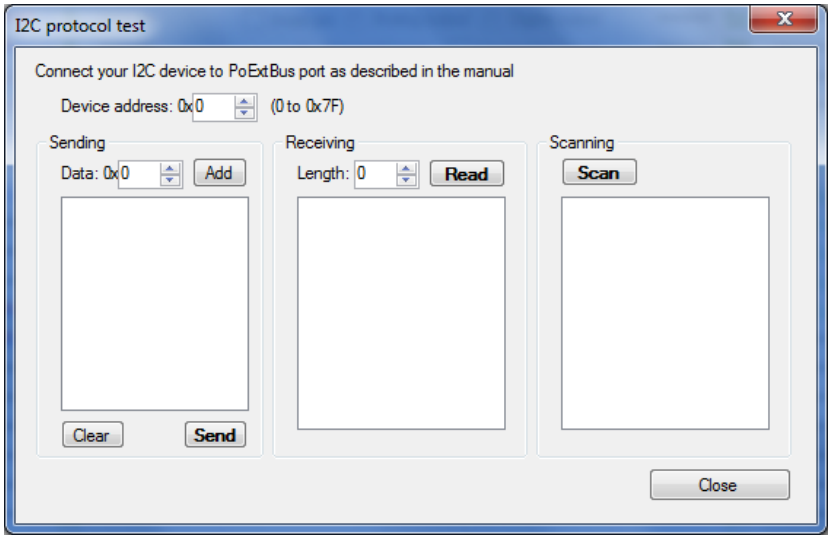


Figure 25: I²C protocol test dialog

Applicable PoKeys library commands

I2C bus commands	
I2CStartBusScan I2CReadBusScanResults	Initiates I2C bus scan for devices Reads results of I2C bus scan
I2CStartRead I2CGetReadStatus	Initiates read from I2C device Read status of reading from I2C bus and retrieve data is

	successfull
I2CStartWrite I2CGetWriteStatus	Writes data to I2C bus Read status of writing to I2C bus

8.16. 1-wire

1-Wire is a device communications bus system designed by Dallas Semiconductor Corp. that provides low-speed data, signaling, and power over a single signal. 1-Wire is similar in concept to I²C, but with lower data rates and longer range. It is typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments.

PoKeys56E devices support communication with 1-Wire slave devices (without parasitic power supply), connected to the pin 55 with external pull-up resistor (of approximately 5 kΩ).

Protocol can be tested via PoKeys configuration software. Click on Peripherals > 1-Wire bus test... The following dialog appears.

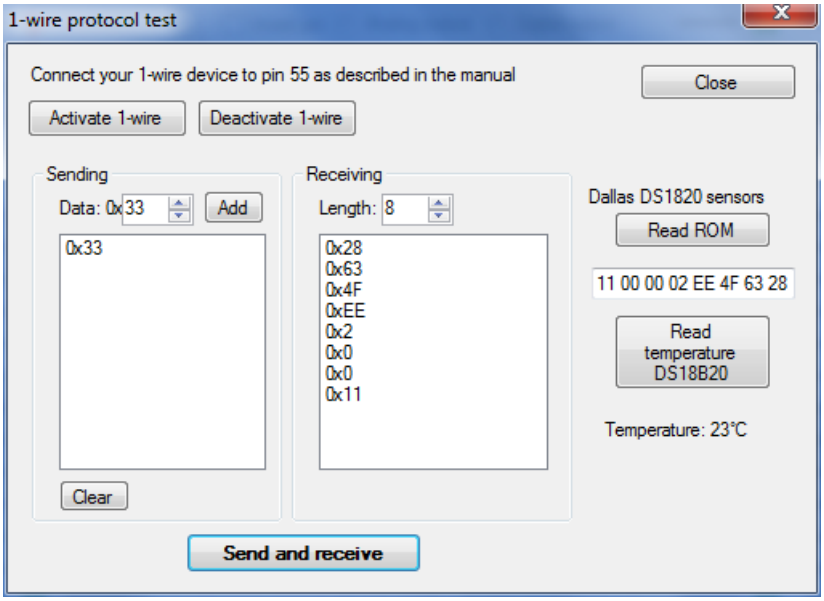


Figure 26: 1-Wire protocol test dialog

Applicable PoKeys library commands

1-wire device configuration and status commands	
prot1WireSetStatus prot1WireGetStatus	Activate or deactivate 1-wire bus support on pin 55 Retrieves 1-Wire bus activation status
prot1WireStartWriteAndRead prot1WireGetReadStatus	Initiates write and read process to/from 1-wire device Read status of write and read command and retrieve data if command was executed

8.17. Sensors

In order to automate the procedure of starting the measurements and reading the results, PoKeys supports up to 10 sensors on I²C bus, up to 10 1-wire sensors and up to 7 analog sensors. The configuration dialog is available through the menu Peripherals>Sensors.

I²C sensors

To setup a I²C sensor, click on an empty ('Inactive') entry in the list, select the sensor's address (the address can be displayed by clicking the 'Scan' command button. A list of addresses of all active I2C devices will be displayed), sensor type and the refresh period. Click Save. Sensor is activated by Sending the configuration to device from the main PoKeys configuration application window. The following list of sensors is supported:

- LM75 temperature sensor for the temperature range -55 °C to +125 °C with the resolution of 0.5 °C. The sensor has configurable address and up to 8 sensors can be connected.
- SHT21 temperature and humidity sensor for the temperature range -40 °C to +125 °C with the resolution of 0.01 °C and air relative humidity in range 0 to 100 %. Due to specifics of the PoKeys sensor list entries, this sensor is represented as temperature sensor and as relative humidity sensor separately. Two entries are needed to read both air temperature and humidity.

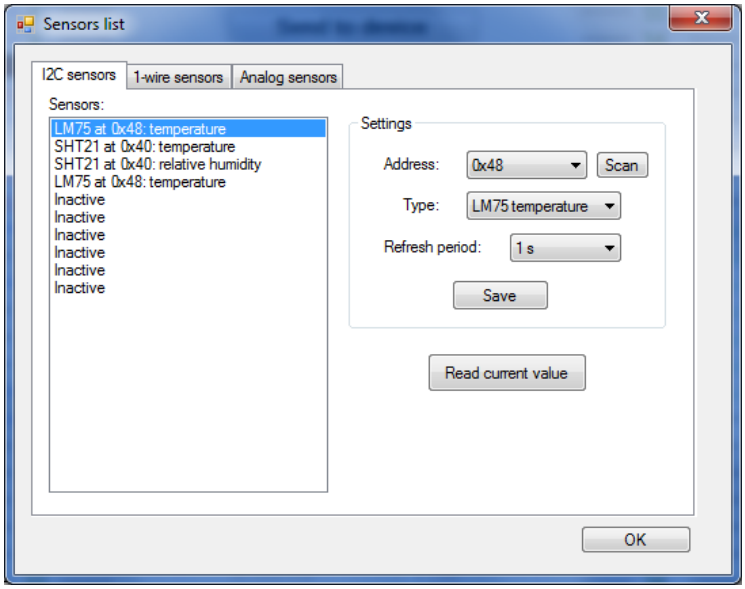


Figure 27: Configuration of I²C sensors

1-wire sensors

To setup a 1-wire sensor, click on an empty ('Inactive') entry in the list, enter the sensor's 64-bit ID sequence and select the refresh period. If there is only one 1-wire sensor present on the 1-wire bus, the ID sequence can be read by clicking on the 'Read curent' command button. Multiple sensors can be setup this way – one sensor at a time. When all sensors are then connected to the bus, the PoKeys will identify each with the help of ID sequence. The following list of 1-wire sensors is supported:

- DS18B20 temperature sensor for the temperature range of -55 °C to +125 °C with the resolution of 0.0625 °C. As each sensor has its own unique sequence ID, up to 10 sensors can be connected to PoKeys.

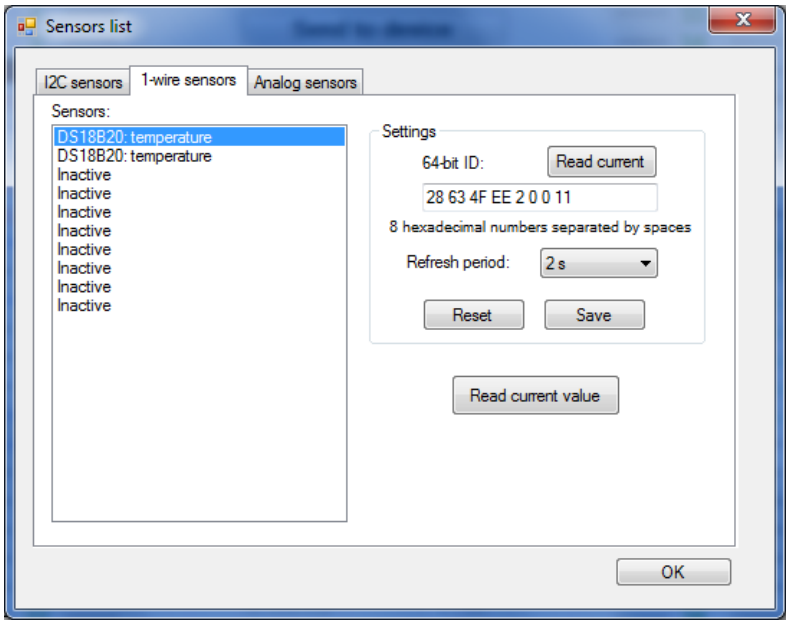


Figure 28: Configuration of 1-wire sensors

Analog sensors

In addition to I²C and 1-wire sensors, PoKeys suports also analog sensors that are connected to analog voltage inputs on the PoKeys board. For each analog sensor, a 2-parameter linear transformation is supported. The following formulae is used:

$$u = AD_{val} * \frac{A_{gain}}{4096} + A_{offset}$$

Where AD_val is a measurement of the analog-to-digital converter (a value between 0 and 4095), A_{gain} is gain (16-bit integer number) and A_{offset} is result offset (16-bit integer number). The value of u is a integer number that gets divised by 100 for the display (the temperature of 15.58 °C is represented by $u = 1558$. A gain of 330 therefore gives the true voltage on the analog input pin.

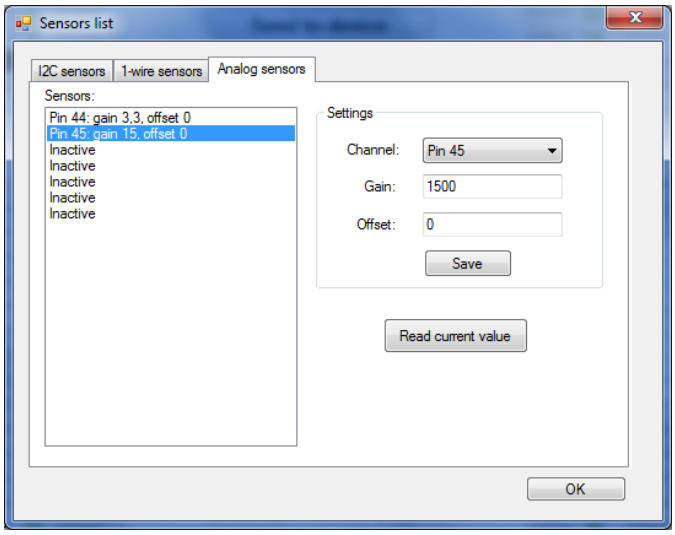


Figure 29: Configuration of analog sensors

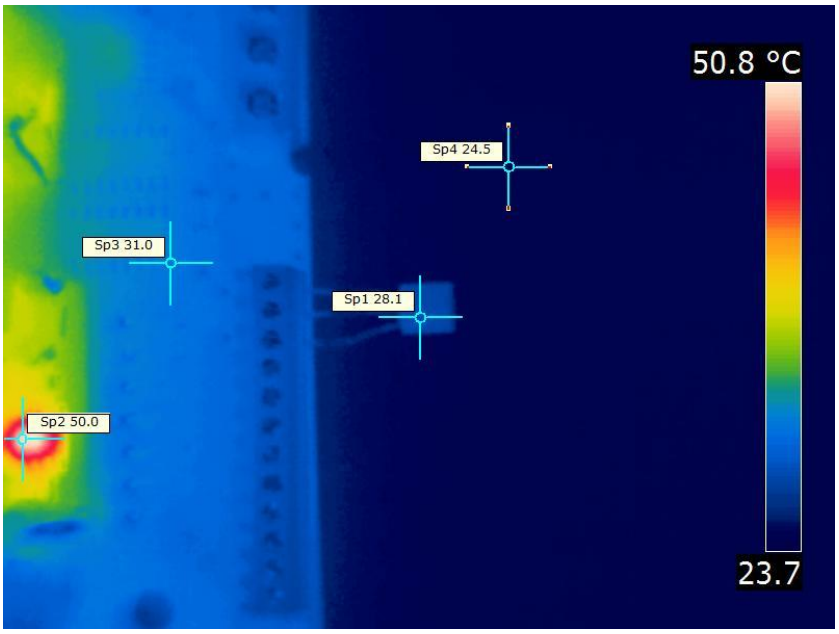


Figure 30: Thermal image of a temperature sensor being connected directly to a circuit board. The thermal conductivity of the sensor's leads causes the sensor to register higher temperature than the ambient real temperature. Use properly longer connections between sensor and sensor host board for the accurate readings

Applicable PoKeys library commands

Sensor configuration	
GetSensorSetup SetSensorSetup	Retrieve the sensor setup and current sensor value Send the sensor setup to device
Read sensors (PoTLog27 specific commands)	
GetAllSensorIDs GetAllSensorValues GetAllSensorValuesString	PoTLog27 command to retrieve 64-bit IDs of all connected sensors PoTLog27 command to retrieve the values of all sensors PoTLog27 command to retrieve the values of all sensors in one string

8.18. PoKeys56E

PoKeys56E is a network type of PoKeys device. The device can be connected to Ethernet 10/100 network with standard RJ-45 cable. Battery holder is for future upgrades only and no battery should be inserted into the holder.

By default, the device is set to use the DHCP functionality of the network router. User can later turn on or off the DHCP support. If DHCP is not needed/wanted, fixed IP address can be defined for the device. To set the network settings of the device, go to Device menu and click Network device settings...

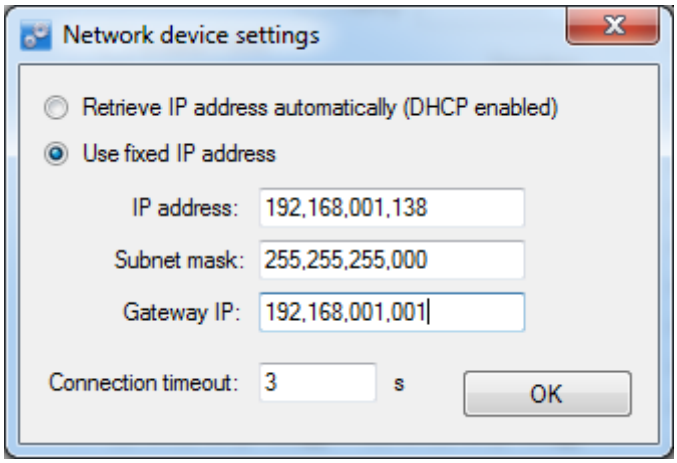


Figure 31: Device network settings

The device communicates using TCP and UDP port of 20055. **Please ensure the firewall settings allow communication with this port. Also, please make sure that your network card (which you have connected PoKeys56E to) has an IP address assigned with the subnet mask 255.255.255.0 (check it in IPv4 settings in your system).**

By default, PoKeys56E device is configured to close the connection with the host after 3 seconds of inactivity. This connection timeout value can be set in the dialog, shown in Figure 31.

Connecting to PoKeys56E device for the first time

1. Connect PoKeys56E with your Ethernet network using the RJ-45 cable
2. Connect power supply for PoKeys56E board
    a. If your network uses DHCP to assign IP addresses to each device, PoKeys56E will be assigned a new IP address automatically
    b. If your network uses fixed IP addresses, PoKeys56E device will wait for discovery packet from the PoKeys configuration software (during this time, LED will blink). In this process, temporary IP address from the same subnet will be assigned to PoKeys56E board.
3. Open PoKeys configuration software and wait for devices to be detected
4. If different network configuration is needed, select device and click on Configure button to select new settings.
5. Click on Connect button to connect to PoKeys56E device and start configuring

Device discovery

If device IP address is set as fixed, use can connect directly to PoKeys56E device. Otherwise, UDP discovery packet should be send as UDP broadcast packet (for details, please see the protocol specification document). All PoKeys56E devices that receive this packet, respond with their current IP address and serial number.

Default settings

DHCP:	**enabled**
Port:	20055
Security:	Full access

After receiveing the UDP discovery packet and if the DHCP server is not available, PoKeys56E will use the temporary address of x.x.x.250, where x.x.x is the subnet address (with 255.255.255.0 subnet mask). This enables user to reconfigure the device with the proper IP address. To do this, click on the Configure button.

Connecting to device in other network

When the device is not detected automatically (either there is a firewall blocking the UDP broadcast messages or the device is not in the same network as a computer), custom IP address of the device can be entered by clicking on the 'Network settings... ' button. The following dialog appears.

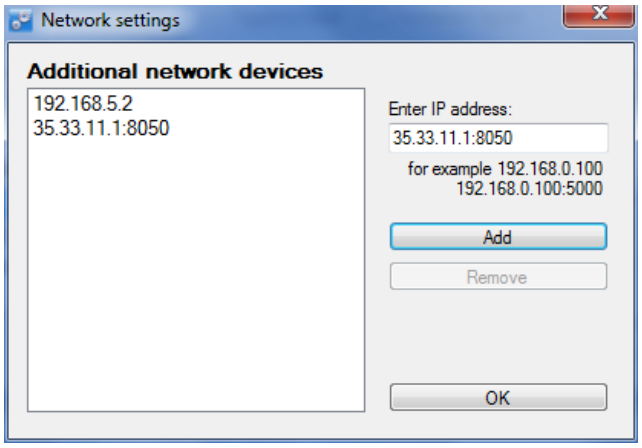


Figure 32: Additional network settings

IP address of the device can be entered in the text box on the right and added to the list by clicking the button 'Add'. The list of additional devices is saved on application exit.

Security

Due to exposed nature of a network device, an authentication mechanism was implemented in PoKeys56E that allows three levels of access rights:

- Full access (default): the device is fully accessible from the network
- Read-only access: unauthorized users are allowed only to fetch data from the device, while an authenticated users can acccess all functions of the device

- Full lock: unauthorised users can not neither read or write to the device. A user password is required to unlock access.

The security is set up in PoKeys configuration software – on the Device menu, click Set device security... The password can contain any character and can be up to 32 characters long.

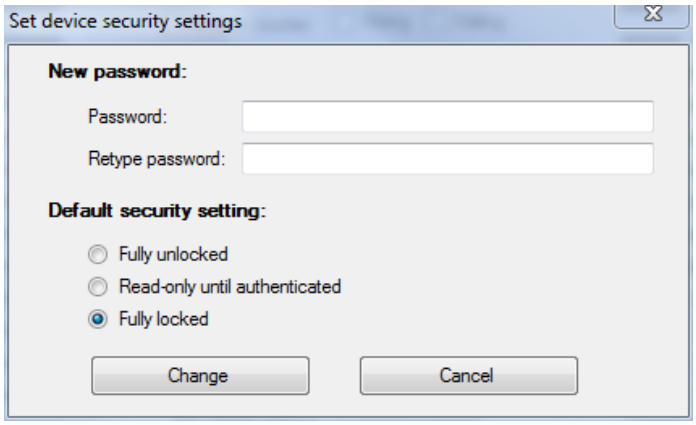


Figure 33: Device security settings window

Applicable PoKeys library commands

PoKeys56E user authorization commands	
AuthorizeUser	Authorizes user on the device - changes the security level to the specified value
GetSecurityStatus	Reads the current security status of network device

Web interface

PoKeys56E devices can be monitored through the simple web interface (that is already enabled by default). The interface can be disabled or configured in dialog accessible via menu Device->Web interface configuration. The following dialog appears

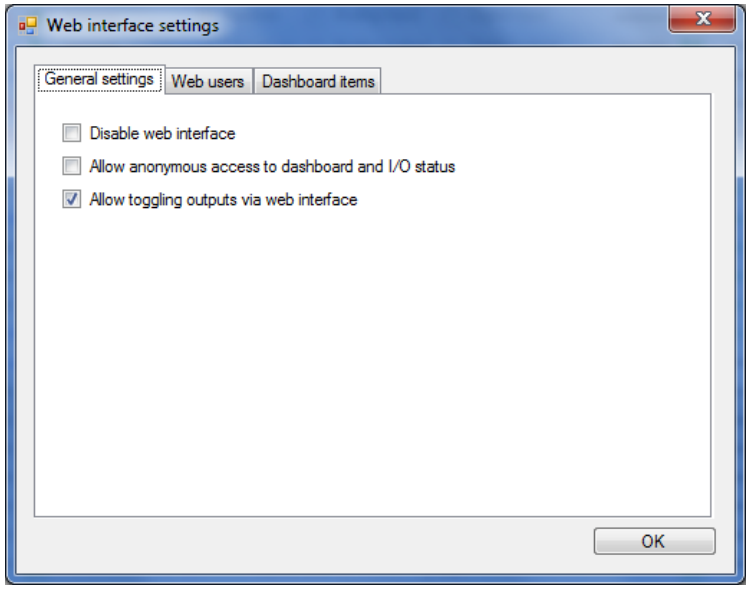


Figure 34: Web interface settings – general settings

The dialog presents the following options on the 'General settings' tab:

- Disable web interface: check this field to disable web interface.
- Allow anonymous access to dashboard and I/O status: if this field is checked, users can access web interface directly without entering user name and password.
- Allow toggling outputs via web interface: if this field is checked, users can toggle the pins that are setup as outputs. If this field is unchecked, users are only presented with the status of each pin.

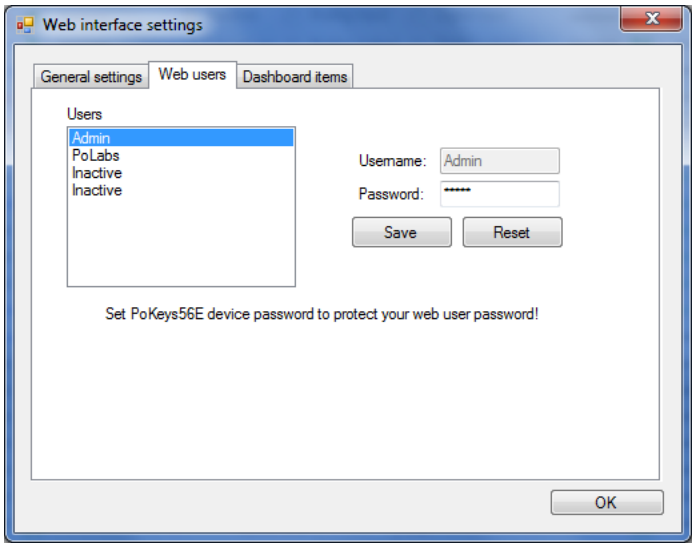


Figure 35: Web interface settings – web users setup

On the 'Web users' tab, usernames and passwords can be configured for up to 4 users (first user is fixed and named 'Admin' with the factory set password 'root0'). Usernames (except for Admin) and passwords can be up to 8 (ASCII) characters long.

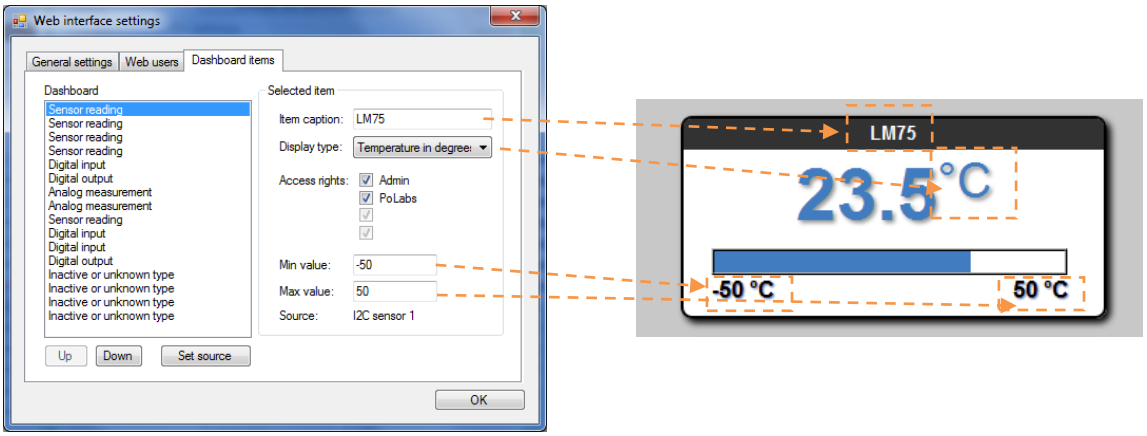


Figure 36: Web interface settings – dashboard items

This page of Web interface settings is used to defined which items will be displayed on the Dashboard page of web interface (see below). PoKeys supports up to 16 items on dashboard screen. Each item displays the status of one sensor, one input or one output.

To set up a new dashboard item, select a 'Inactive or unknown type' entry in dashboard list and press 'Set source' command button. In a list that appears, select either 'Digital input', 'Digital output' or 'Sensor'. For the digital inputs and outputs, a dialog for entering pin number will appear, while for the sensors, a sensor selection dialog will appear. After selecting either digital input/output or sensor source, additional options will be available. For each item in the list, a 8 character item caption can be assigned. Available display types depend on the selected item data source (see Table 4 for details). For display types that include a progress bar, additional min and max value for the progress bar can be specified.

Item data source	Supported display types
Digital input	Digital input status
Digital output	Digital output status
	Digital output status with buttons
Sensor	Value in V
	Value in V and bar graph
	Value in mA
	Value in mA and bar graph
	Value in A
	Value in A and bar graph
	Temperature in degrees C
	Temperature in degrees C and bar graph
	Relative humidity in %
	Relative humidity in % and bar graph

Table 4: Display types for different item data sources

Access rights settings enables administrator to select which users will be able to display the sensor readings. This is intented mainly for display items that can change outputs.

To open the interface, use your internet browser and type in the IP address of the PoKeys56E board. After entering the username and password, the following page will be displayed:

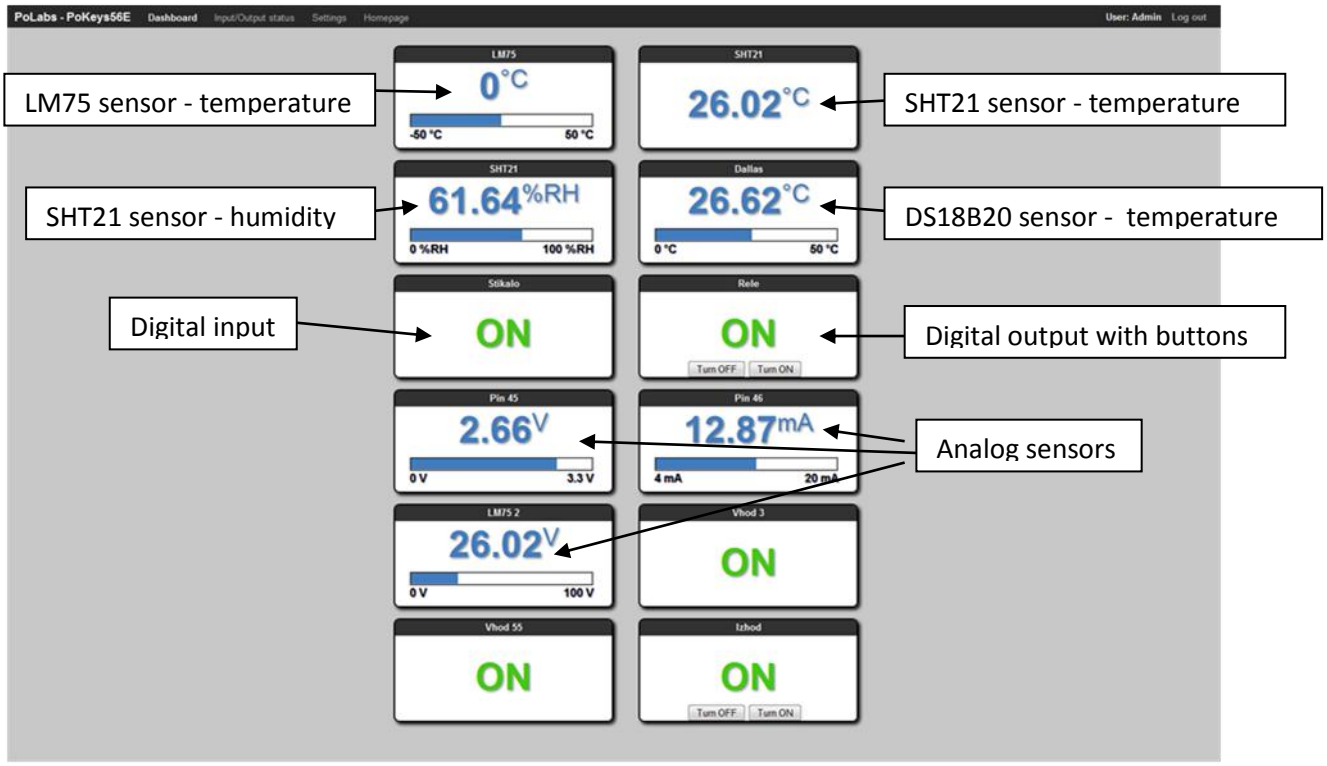


Figure 37: PoKeys56E dashboard

Developers can access status of the inputs by fetching the file /devStat.xml from PoKeys56E board.

Applicable PoKeys library commands

Web interface configuration commands	
GetWebSettings	Retrieve web interface settings (enable/disable interface, anonymous access, outputs enable) from the device
SetWebSettings	Send web interface settings (enable/disable interface, anonymous access, outputs enable) to device
GetWebUserData	Retrieve user name and password for the selected web user account
SetWebUserData	Send user name and password for the selected web user account
GetDashboardItem	Retrieve the dashboard item setup
SetDashboardItem	Send the dashboard item setup

Modbus

PoKeys56E supports slave (server) operation of Modbus TCP communication protocol. Modbus TCP compatible devices on the network can read the values from the device and set the outputs. To elevate the security, user can define which peripherals are accessible via Modbus TCP.

Modbus TCP uses TCP protocol on port 502 (default), which can be changed in Modbus settings (accessible from the menu Device – Modbus configuration...). The Modbus TCP connection is disconnected after 3 seconds of inactivity (this default value can be changed in the Modbus settings).

Discrete inputs/outputs

Supported operations:

0x01: Read coils
0x02: Read discrete input
0x05: Write single coil
0x0F: Write multiple coils

Address (0-based)	Access (R – Read, W – Write)	Description
0-54	R/W	55 pin inputs/outputs
1000-1127	R	Matrix keyboard inputs
1400-1527	W	I2C Matrix keyboard LED
1600-1727	R/W	LED matrix
2000-2079	R/W	PoExtBus
2000		Device 10 – Output H
2001		Device 10 – Output G
...		
2079		Device 1 – Output A

Registers

Supported operations:

0x03: Read holding register
0x04: Read input register
0x06: Write single register
0x10: Write multiple registers

Address (0-based)	Access (R – Read, W – Write)	Description
10-16	R	Analog inputs
20-45	RW	Encoder counter values (lower 16-bit)
100-154	RW	Digital counter values
200-213	RW	PWM
200,201		PWM period (MSB first)
202,203		PWM duty1 (MSB first) – pin 22
...		
212,213		PWM duty6 (MSB first) – pin 17
300-304	RW	PoExtBus
400-453	R	Sensors (32-bit values, LSB first)
500-579	RW	LCD buffer

590	W	LCD configuration (0=disabled, 1=primary or 2=secondary) - writing to this register will re-init and clear the LCD
591	W	Number of rows (lower byte) and number of columns (upper byte) of the LCD module
592	W	Not used
593	W	Clear LCD (both bytes = 0xAA)
600	R	Tick counter (lower 16-bit)
700-751	R[W]	Digital encoder values (32-bit values, LSB first) - any write to these registers causes the reset of the encoder value to 0
800-909	RW	Digital counter values (32-bit values, LSB first)

PoExtBus channel mapping:

Address (0-based)	Register description															
300	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 10								**Device 9**							
301	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 8								**Device 7**							
302	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 6								**Device 5**							
303	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 4								**Device 3**							
304	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 2								**Device 1**							

where

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Modbus word bit**
A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H	PoExtBus device bit mapping
Device 10								**Device 9**								

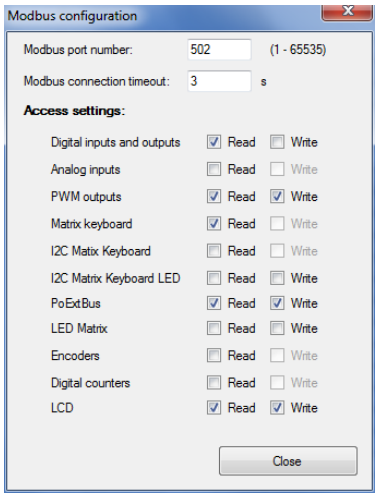


Figure 38: Modbus configuration

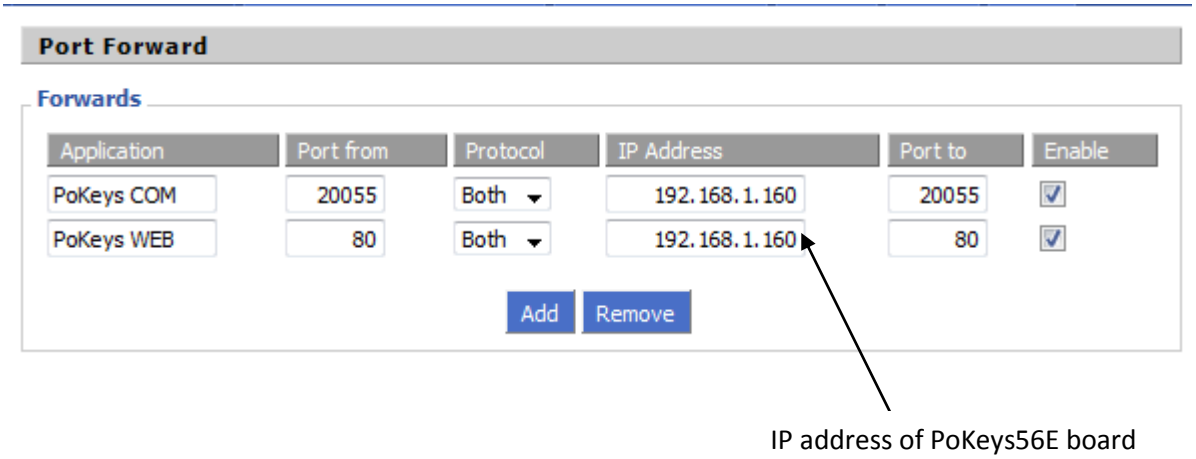
Applicable PoKeys library commands

Modbus configuration commands	
GetModbusSettings SetModbusSettings	Retrieves modbus settings Set modbus settings

Connecting to PoKeys56E across the internet

To access the PoKeys56E from the internet, connect the PoKeys56 board to your router and configure the following port forwarding (consult router manual for instructions on port forwarding):

- For the WEB interface, forward the port 80
- For the use of the PoKeys software, forward the port 20055

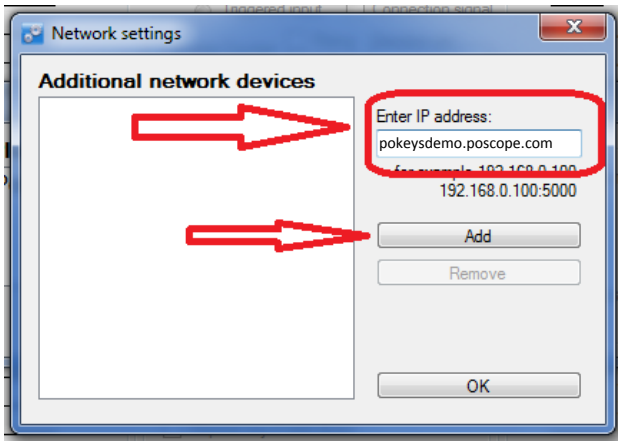


Using PoKeys software with PoKeys56E device across the internet

Suppose that the PoKeys56E is at the *pokeysdemo.poscope.com* (change this address to the address of your own PoKeys56E device).

Open PoKeys software and click on 'Network settings'.

The following dialog will appear. Enter the address of the PoKeys56 board in the field on the right and click on 'Add'.



Then click on 'OK'. If the connection was established successfully, PoKeys56E will be shown in the list of the devices.

8.19. Reporting data to network server with PoKeys56E device

PoKeys56E devices can automatically report sensor values to various network servers using the HTTP POST, HTTP PUT or text-only protocols.

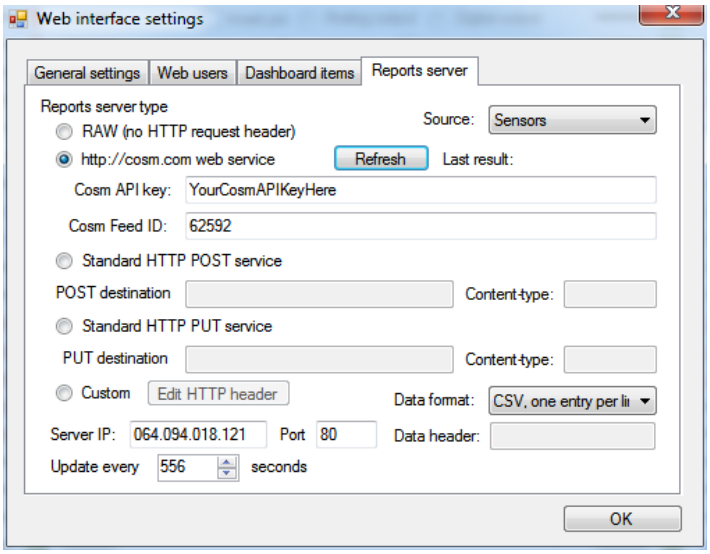


Figure 39: Reports server settings for Cosm service

To use this reporting feature, user must specify request type (HTTP POST/PUT, Cosm, custom HTTP header or raw), server IP and update rate.

The request header is constructed of two parts – HTML header and data header, divided by double new line character (\n). Example header (Cosm.com web service):

PUT /v2/feeds/62592.csv HTTP/1.1 Host: api.cosm.com X-ApiKey: CAb3XX634daSAKxZc3M0M3I1NWVZVT0g User-Agent: PoKeys56E Content-Type: text/csv Content-Length: 010 Connection: close Test1234,-15000.00	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device Data is inserted at the end of the packet

The above example is specified as

```
PUT /v2/feeds/62592.csv HTTP/1.1
Host: api.cosm.com
X-ApiKey: CAb3XX634daSAKxZc3M0M3I1NWVZVT0g
User-Agent: PoKeys56E
Content-Type: text/csv

```

The extra new line at the end is essential.

More customized header:

POST /myScript.php HTTP/1.1 Host: api.cosm.com User-Agent: MyCustomDeviceAgent Content-Type: text/csv Content-Length: 010 Connection: close MyData: Test1234,-15000.00	The HTTP header without 'Connection' and 'Content-length' tags must be provided by the user The 'Connection' and 'Content-length' tags are automatically inserted by PoKeys56E device

The above example must be specified as

```
POST /myScript.php HTTP/1.1
Host: api.cosm.com
User-Agent: MyCustomDeviceAgent
Content-Type: text/csv

MyData:
```

Again, an extra new line character in fifth line is essential.

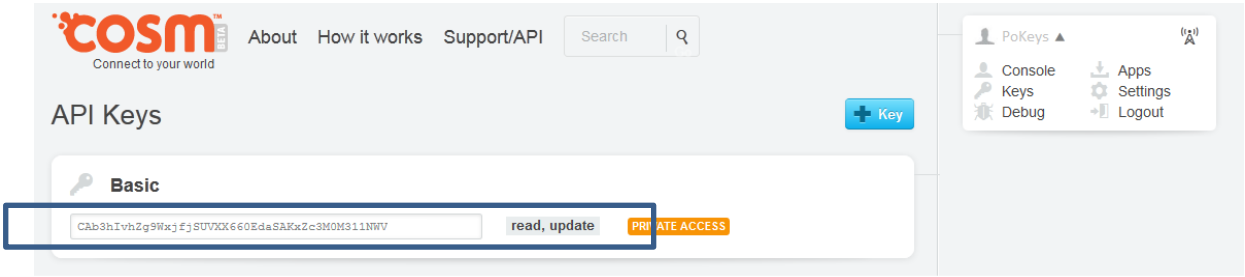
Total length of final header and data is limited to 350 bytes.

Setup for http://cosm.com/ service

http://cosm.com/ is an on-line database service allowing users to connect sensor-derived data (eg. energy and environment data from objects) to the Web and to build their own applications based on that data.

PoKeys56E devices feature a direct support for the http://cosm.com/ service. To configure PoKeys device for http://cosm.com/ service, follow these steps:

1. Sign-up for a free account at http://cosm.com/
2. Create a new empty feed and save its Feed ID
3. On the http://cosm.com/ homepage, go to Keys and create a new API key for a newly created feed that has read and update privileges



4. Open Device > Web interface settings dialog and go to 'Reports server' tab. Check 'http://cosm.com/ web service' option.
5. Enter your Cosm API key, created in step 3, and Feed ID from step 2, leave other fields with default values
6. Select the update rate at the bottom.
7. Click OK and click Send to device button.

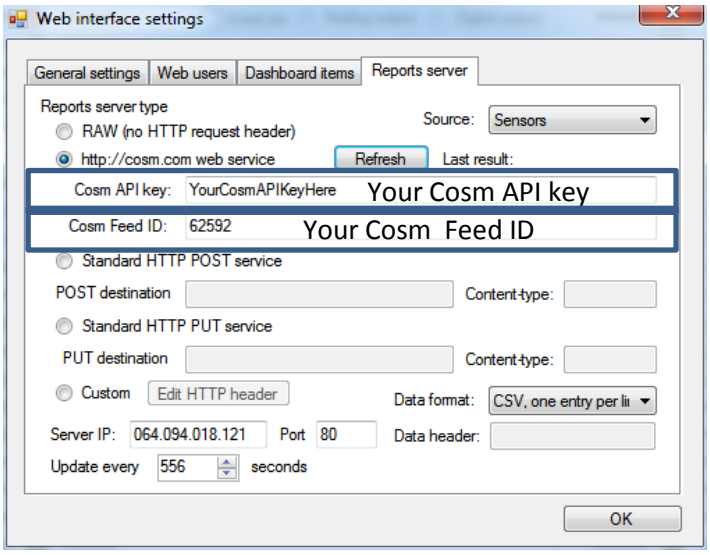


Figure 40: Cosm settings page

8. Go back to Device > Web interface settings to page 'Dashboard items'. Define entries as described in the 'Web interface' chapter of this manual. To enable uploading of the dashboard item to the http://cosm.com/ service, select 'Web report' as the user. The 'Item caption' field is used to identify the datastream in the selected Cosm feed.

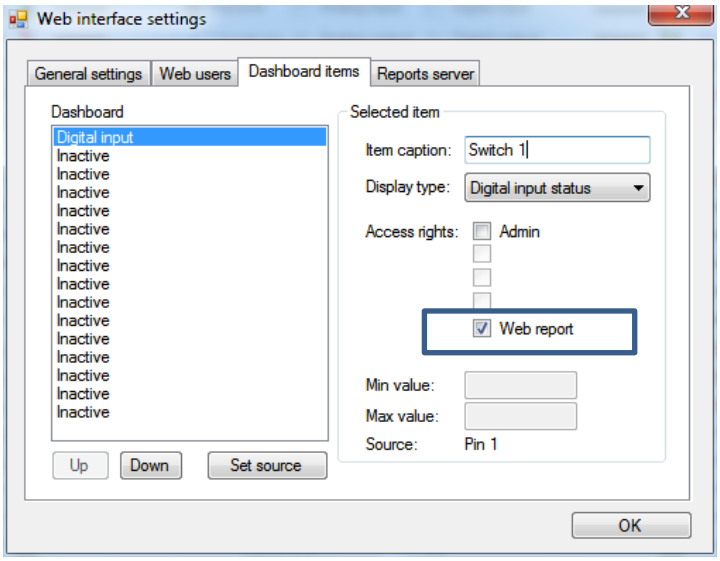


Figure 41: Item configuration for the Cosm service

9. Save the settings again by clicking 'Send to device' button.
10. Make sure the PoKeys device has properly configured network settings and that it is connected to the internet
11. After a update interval, check the status of the Cosm updates by clicking 'Refresh' button in the Cosm settings page.
12. Open your Cosm feed – you should see the recorded data in the graphs

Figure 42: Cosm Feed overview page

Setup for standard HTTP POST or PUT data upload

Let's assume that the user wants to send sensor data to his server script that accepts POST method for data upload. The user's server script is available at the address www.userdomain.com/PoKeysDataUpload.asp. The user also wants that the data in the POST stream to have the HTTP type of 'text/plain' and preceeded with the string 'MyData: '. The data should be transferred every 10 minutes. The settings are displayed in Figure 43.

To setup PoKeys to use this settings, the following steps must be taken (enter text without ' quotes):

1. Open Device > Web configuration menu and switch to 'Reports server' tab.
2. Select 'Standard HTTP POST service' option
3. Enter 'www.userdomain.com/PoKeysDataUpload.asp' in the POST destination field
4. Enter 'text/plain' in the 'Content-type' field.
5. Enter 'MyData: ' in the 'Data header' field
6. Enter the server's IP address and port number in the fields below.
7. Enter 600 in the update time field
8. Click OK and click Send to device button.
9. Go back to Device > Web interface settings to page ‘Dashboard items’. Define entries as described in the ‘Web interface’ chapter of this manual. To enable uploading of the dashboard item to the web server, select ‘Web report’ as the user. The ‘Item caption’ field is used to identify the datastream.
10. Save the settings again by clicking 'Send to device' button.
11. Make sure the PoKeys device has properly configured network settings and that it is connected to the internet

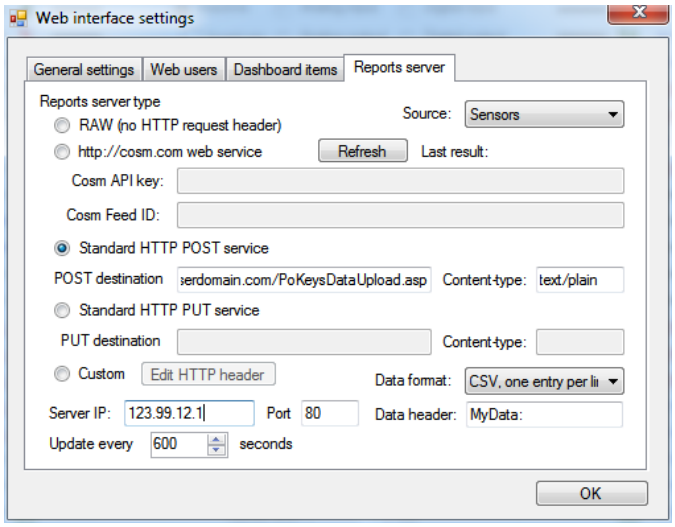


Figure 43: Example of HTTP POST service setup

8.20. Changing User ID number

Users can freely assign their own User ID number that represents a specific PoKeys device (enables distinguishing between different PoKeys devices in case there is more than one connected to a single host PC). To change the User ID number, go to ‘Device’ > ‘Change user ID’ menu. Simply enter any number between 0 and 255, and click the 'Change user ID' button.

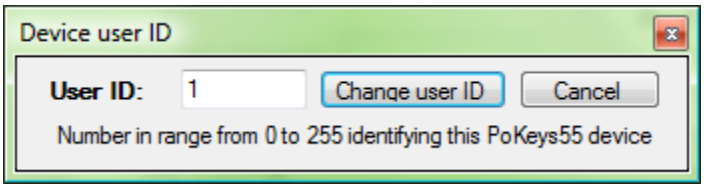


Figure 44: Device user ID dialog

8.21. Saving current configuration to file

To save the current configuration to a file, go to ‘File’ > 'Save' menu and select a new filename. To reload a saved configuration from a file, go to ‘File’ > ‘Open’ menu and select the appropriate file. To transfer new settings to the device, click on the ‘Save to device’ button.

9. PoTLog27 firmware

PoKeys56U and PoKeys56E devices support an alternate firmware that allows the connection of up to 27 DS1820 or DS18B20 temperature sensors (one sensor per pin, pins 1-27 only). Firmware automatically detects the sensors on each power-up and starts reading the temperatures. Temperatures of both types of temperature sensors are read at the rate of 1 measurement per 800 miliseconds (around 1,25 measurements per second) at the resolution of 12-bit (approx. 0,06 °C).

To activate PoTLog27 firmware, go to menu Device > Update firmware to PoTLog27... and wait for the update to finish. After the successfull update, the device will reboot into the PoTLog27 firmware.

To check the configuration and current temperatures, go to menu Peripherals > PoTLog status... The following dialog will appear, showing the list of all detected sensors and their temperatures.

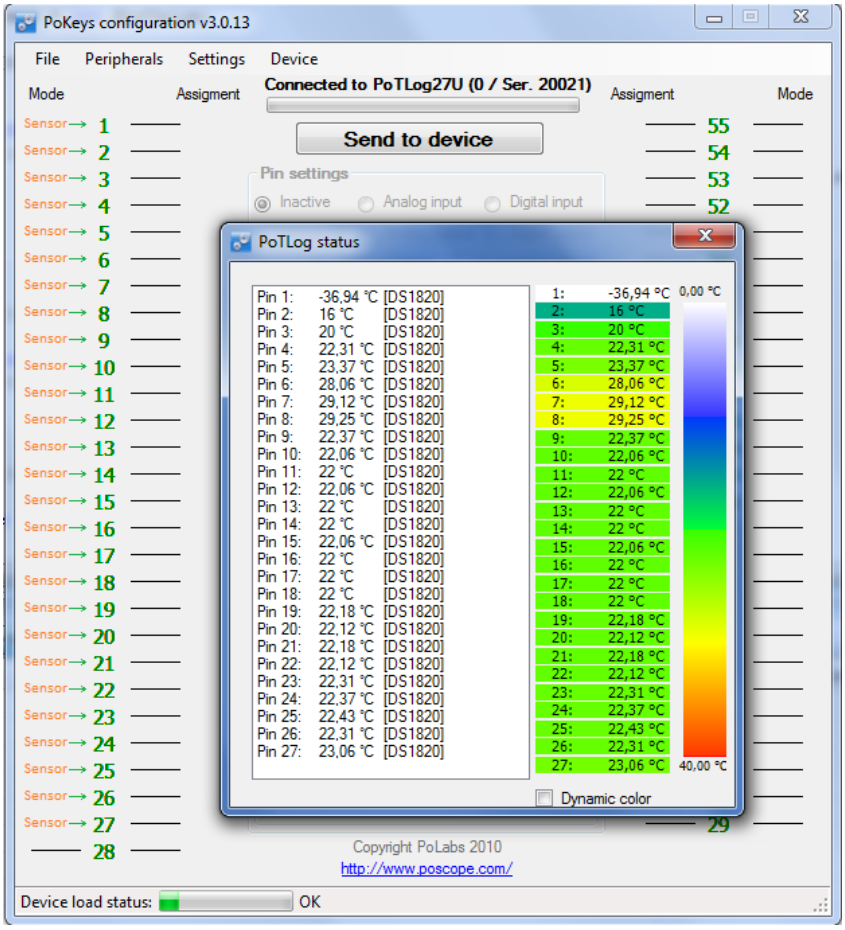


Figure 45: PoTLog status dialog

Note: the following functions are disabled in PoTLog27 mode:

- Encoders
- Macros
- Triggered key mapping and delayed/repeated key actions
- PWM outputs
- Joystick analog to digital mapping
- Connection signal
- PoExtBus
- Additional 1-wire devices support
- Custom device name

To access the temperature measurements, these commands are available in PoKeys communication dll:

GetAllSensorValuesString

Retrieves all measurements in one string. Each measurements is separated with | sign.

```
string GetAllSensorValuesString()
```

Arguments:

none

Remarks:

none

GetAllSensorValues

Retrieves all measurements in one structure.

```
bool GetAllSensorValues(ref sPoTLogDevice logDevice)
```

Arguments:

none

Remarks:

C++ alternative: COM_GetAllSensorValues(IntPtr logDevice)

GetAllSensorIDs

Retrieves all sensor 64-bit IDs in one structure.

```
bool GetAllSensorIDs(ref sPoTLogDevice logDevice)
```

Arguments:

none

Remarks:

C++ alternative: COM_GetAllSensorIDs(IntPtr logDevice)

Example code (C#):

```
sPoTLogDevice logger = new sPoTLogDevice();

MyDevice.GetAllSensorValues(ref logger);
MyDevice.GetAllSensorIDs(ref logger);

for (int i = 0; i < 27; i++)
{
    byte sType = logger.sensors1Wire[i].sensorID[0];
```

```
    double value = (double)logger.sensors1Wire[i].sensorValue / 100;
    string displayValue = "";

    if (sType == 0x10)
    {
        displayValue = value + " °C";
        listBox1.Items.Add("Pin " + (i + 1) + ":\t" + displayValue + "\t[DS1820]");
    }
    else if (sType == 0x28)
    {
        displayValue = value + " °C";
        listBox1.Items.Add("Pin " + (i + 1) + ":\t" + displayValue + "\t[DS18B20]");
    }
    else
    {
        listBox1.Items.Add("Pin " + (i + 1) + ":\tNo sensor");
    }
}
```

10. Communicating with the USB devices using the console

In the software bundle included with the PoKeys device, there is also a console interface application, which enables command-line style communication with the device (figure 6). To start using the console application, go Start>Run…, type cmd and press Enter. Navigate to the folder, where PoKeysConsole.exe is located (usually C:\Program Files\PoKeys\).

10.1. Supported operations

Enumerate PoKeys55 and PoKeys56U devices

Command line: PoKeysConsole.exe -e

Description: Enumerates and prints out all the detected PoKeys55 or PoKeys56U devices with their User IDs.

Get details of the specific PoKeys55 or PoKeys56U device

Command line: PoKeysConsole.exe –d<user ID>

Description: Prints out the detailed description of the device, i.e. the device's serial number, firmware version and User ID.

Example: PoKeysConsole.exe –d1

Connect to PoKeys55 or PoKeys56U device

Command line: PoKeysConsole.exe –c<user ID>

Description: Before any operation can be executed, host software must connect to PoKeys device, using Connect to PoKeys device operation.

Example: PoKeysConsole.exe –c1

Save current configuration to flash memory

Command line: PoKeysConsole.exe –w

Description: After the settings have been changed, they need to be sent to the device. This is accomplished with via the Save configuration operation. 'Connect to PoKeys55 or PoKeys56U device' operation must be executed before this operation!

Example: PoKeysConsole.exe –c1 –w

Get current pin setting

Command line: PoKeysConsole.exe –g<pin ID>

Description: Prints out the current pin setting. If the pin ID parameter is omitted, settings for all the pins are printed out.

Example: PoKeysConsole.exe –c1 –g10

Set pin setting

Command line: PoKeysConsole.exe –s<pin ID>,<pin function>,+/-

Description: Enables the desired function on the selected pin.

Pin function	Value
Inactive	0
Digital input	2
Digital output	4
Analog input	8
Analog output	16

The last parameter is used to define polarity of digital input and output pins. It must be either + (non-inverted polarity) or - (inverted polarity).

Example: PoKeysConsole.exe –c1 –s10,2,-

Get digital input value

Command line: PoKeysConsole.exe –i<pin ID>

Description: Reads and prints out current digital input value on selected pin.

Example: PoKeysConsole.exe –c1 –i10

Set digital output value

Command line: PoKeysConsole.exe –o<pin ID>,0/1

Description: Sets the digital output to specified value.

Example: PoKeysConsole.exe –c1 –o11,1

Get analog input value

Command line: PoKeysConsole.exe –a<pin ID>

Description: Reads and prints out current analog input value on selected pin. Pin ID must be between 43 and 47, since only these pins support analog to digital conversion.

Example: PoKeysConsole.exe –c1 –a43

Set analog output value

Command line: PoKeysConsole.exe –b<pin ID>,value

Description: Sets the digital output to specified value. Value can be any number between 0 (0 V) and 1023 (3.3 V).

Example: PoKeysConsole.exe –c1 –b43,50

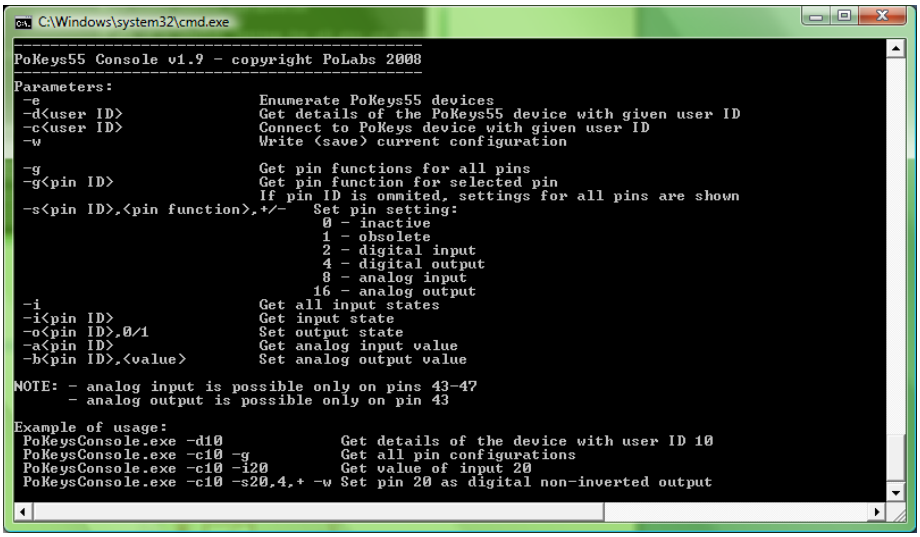
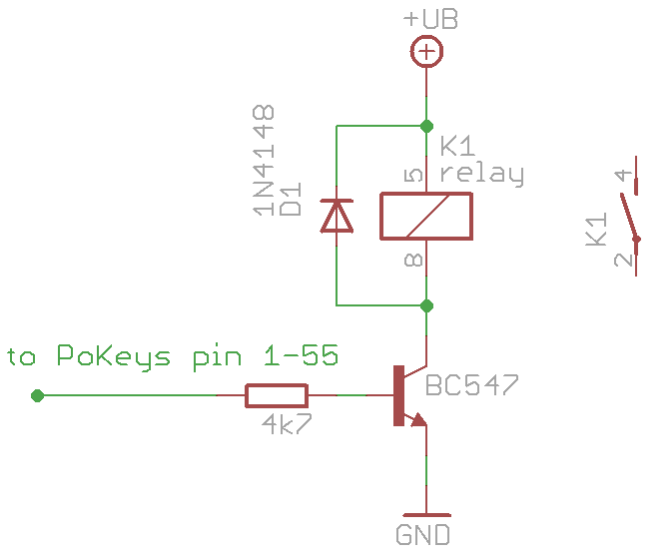

C:\Windows\system32\cmd.exe
PoKeys55 Console v1.9 - copyright PoLabs 2008
Parameters:
-e Enumerate PoKeys55 devices
-d<user ID> Get details of the PoKeys55 device with given user ID
-c<user ID> Connect to PoKeys device with given user ID
-w Write (save) current configuration
-g Get pin functions for all pins
-g<pin ID> Get pin function for selected pin
If pin ID is ommited, settings for all pins are shown
-s<pin ID>,<pin function>,+/- Set pin setting:
0 - inactive
1 - obsolete
2 - digital input
4 - digital output
8 - analog input
16 - analog output
-i Get all input states
-i<pin ID> Get input state
-o<pin ID>,0/1 Set output state
-a<pin ID> Get analog input value
-b<pin ID>,<value> Set analog output value
NOTE: - analog input is possible only on pins 43-47
- analog output is possible only on pin 43
Example of usage:
PoKeysConsole.exe -d10 Get details of the device with user ID 10
PoKeysConsole.exe -c10 -g Get all pin configurations
PoKeysConsole.exe -c10 -i20 Get value of input 20
PoKeysConsole.exe -c10 -s20,4,+ -w Set pin 20 as digital non-inverted output

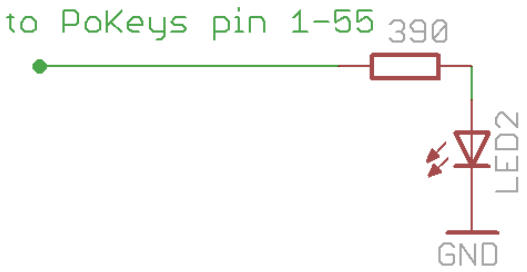
Figure 46: Console application window

11. Connecting common peripherals to PoKeys devices

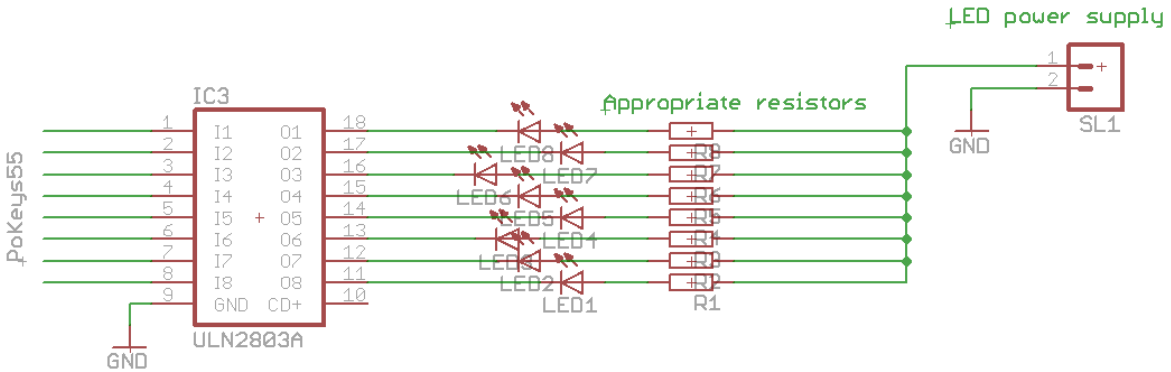
Relays



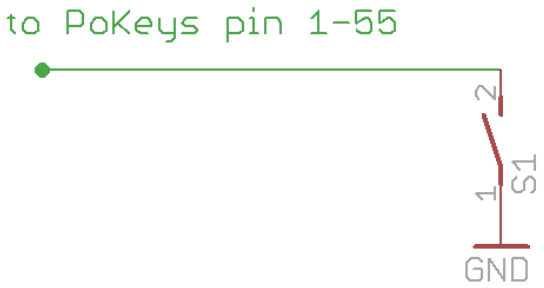
LEDs



High-current LEDs



Switches

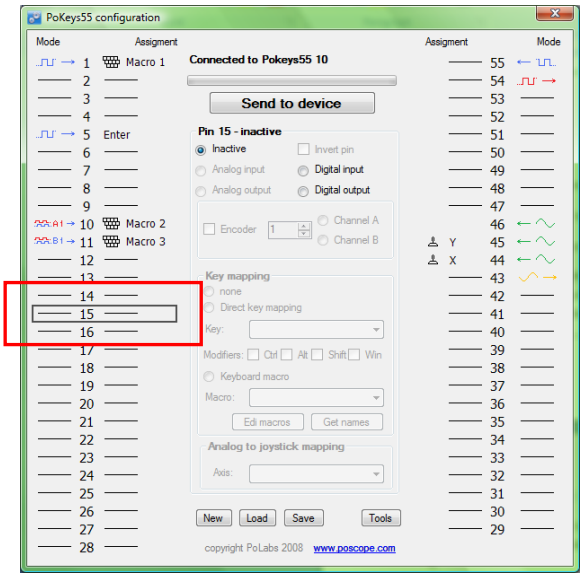


Example: Setting up key mapping

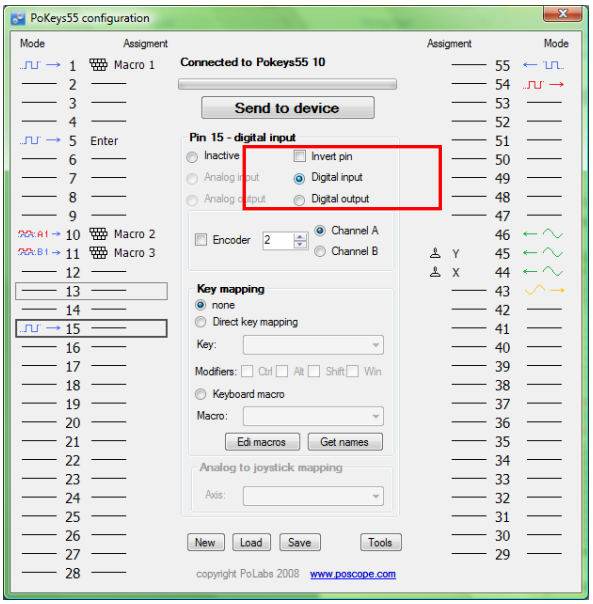
This example shows how easy is to set up a digital input pin for direct key mapping

We will set up a Shift-Escape combination for pin 15.

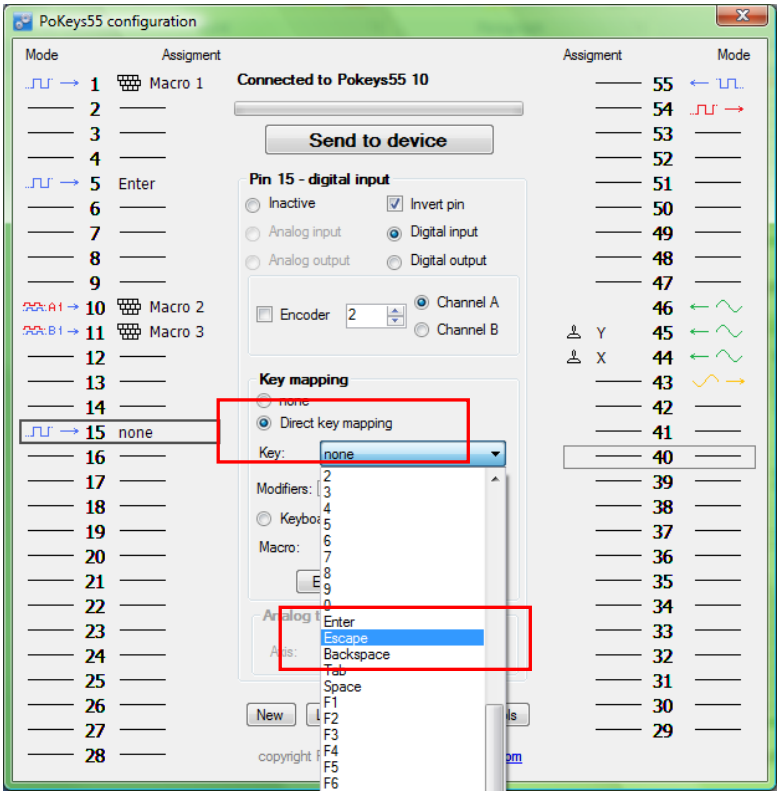
1. Connect a switch to your PoKeys device as shown above
2. Open PoKeys configuration application
3. Select your PoKeys device from drop-down box and click ‘Connect’ button
4. Wait the application to load current configuration from PoKeys device
5. Click the same pin number as you connected a switch to (in this example pin 15)



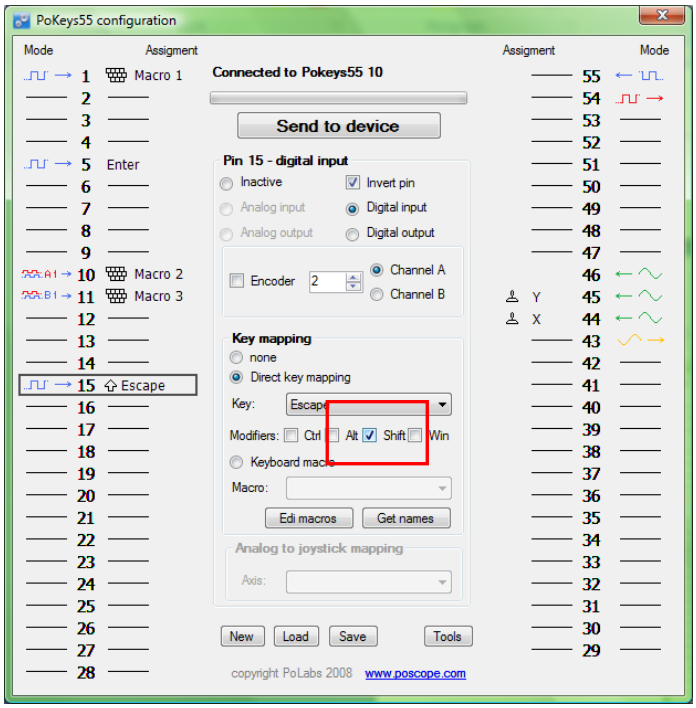
6. Set this pin as digital input



7. Select ‘Direct key mapping’ and from drop-down box select Escape

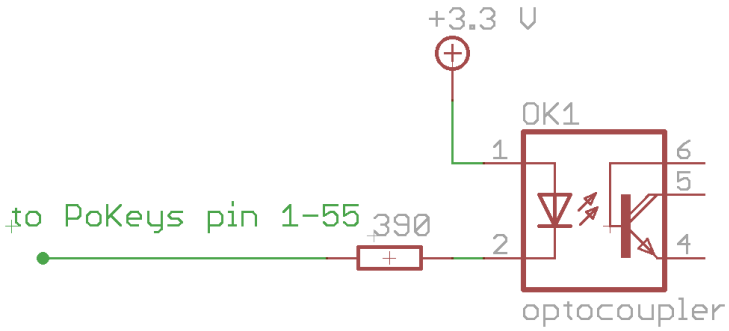


8. Click on the ‘Shift’ checkbox to enable Shift modifier

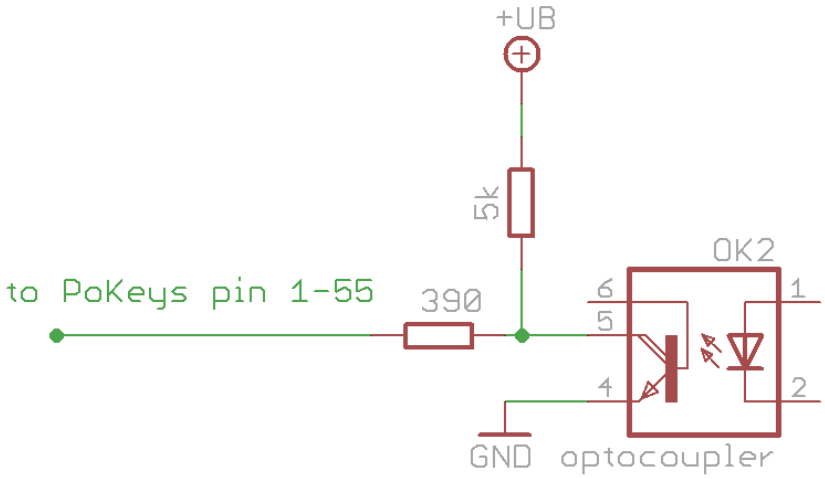


9. Send configuration to device by clicking ‘Send to device button’.

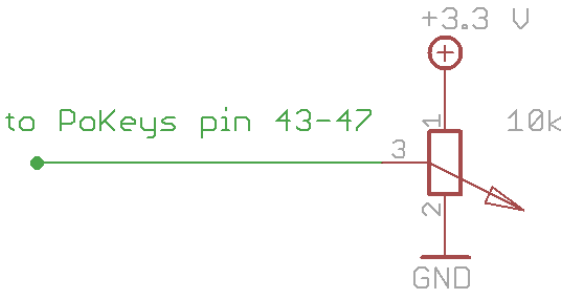
Optocoupled digital output



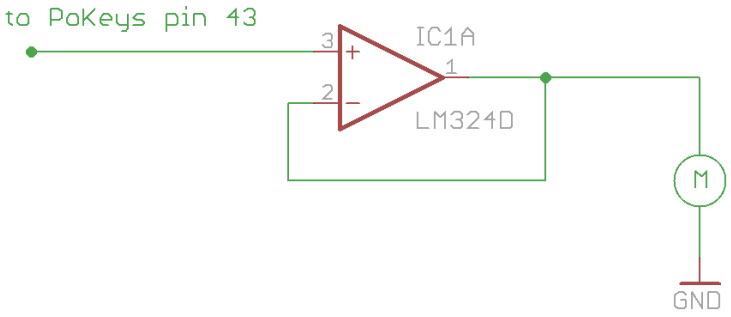
Optocoupled digital input



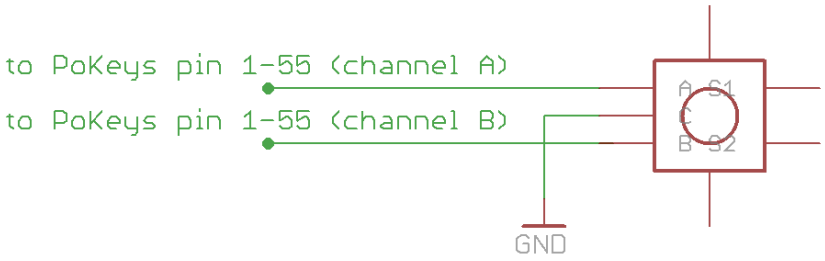
Potentiometers (variable resistors)



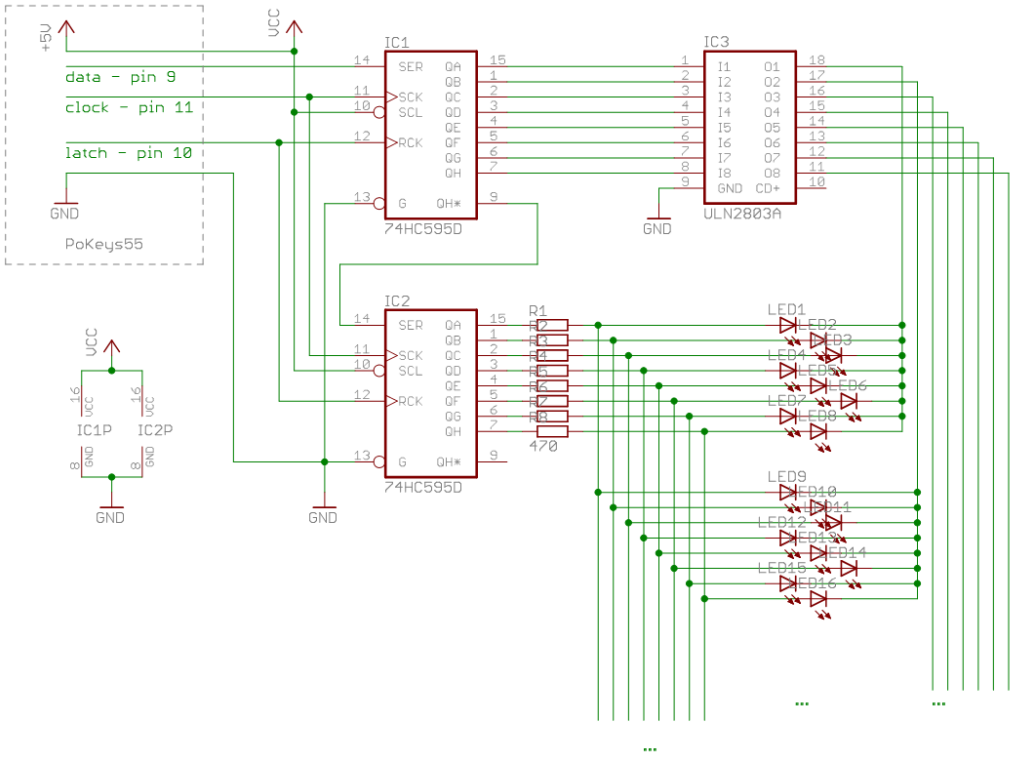
Linear motor control



Rotational encoder switch



LED displays/arrays



12. Quick resetting the device configuration (or recovering bad firmware update)

If configuration editor cannot be used to reconfigure the device because of endless key presses from the device, simple reset procedure should be executed.

1. Disconnect PoKeys device from USB (unplug the power for the PoKeys56E device)
2. Find pin labeled 'RST' on the PoKeys device (otherwise use pin numbered 54)
3. Short this pin to ground (GND) and reconnect the PoKeys device to USB (or reconnect power)
4. Green light should start flashing rapidly and PoKeys device will connect in recovery mode
5. Open PoKeys configuration application
6. PoKeys configuration application should detect PoKeys device in recovery mode.
    a. To reset the device configuration, use the option 'Clear settings'. By clicking this button and confirming your decision on the next dialog, settings will be erased.
    b. To recover from bad firmware update, click 'Recover'.
7. After completing the operation, unplug PoKeys device and disconnect 'RST' pin from ground
8. If resetting the configuration, ensure the device is properly cleared – replug PoKeys device, connect to it and execute Device > Clear settings in device.

PoKeys configuration software is backing up current configuration state (except keyboard macro sequences) on each connection start.

These configuration files can be found in the local application folder (system folder – usually c:\Documents and settings\{username}\Local Settings\Application Data\PoKeys\ on Windows 2000, XP or C:\Users\{username}\AppData\Local\PoKeys\), named backup1.pkc, backup2.pkc and backup3.pkc with backup3.pkc being the oldest configuration.

13. Frequently asked questions

What software must be installed to operate the device?

On first use or when reconfiguring the device, the supplied software must be installed. There are no device drivers needed. They are already supplied with your operating system. Once the device has been configured, the settings are stored on-board. Device can then be freely used on any machine (see requirements for USB HID device driver enabled operating system) without any additional installation.

I misconfigured the device. Now the device starts pressing virtual keys before I can do anything. What can I do?

If you misconfigured the device in such a way that configuration utility cannot be used to repair the configuration, see the section 'Quick resetting the device configuration' in this manual.

How do I connect switch/relay/LED/... to PoKeys device?

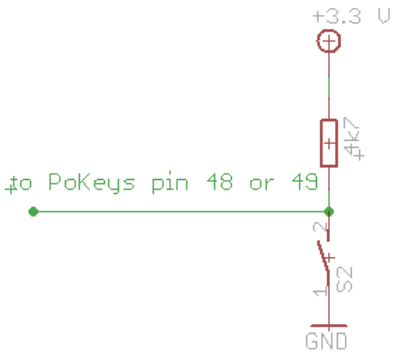
Please see the section 'Connection common peripherals to PoKeys device' in this manual.

I have two (or more) PoKeys devices connected on one system and cannot differentiate the devices to set the configurations.

It is advised that the users assign different UserID numbers to each of the device connected to a system. Please see the section Changing User ID of this manual.

It appears that pins 48 and 49 are floating. What should I do? (PoKeys55 only)

Due to device design, pins 48 and 49 should be equipped with external 5-10 kΩ pull-up resistor as shown bellow.



Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

I have connected a switch to pin 4 and now PoKeys55 is not recognized by the computer anymore.

You must have connected normally-closed switch to pin 4 and therefore connected pin 4 to ground. At boot (connecting PoKeys55 to USB) this means that PoKeys55 is entering system boot and therefore cannot be used from the computer. Please use another pin for normally-closed switches.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

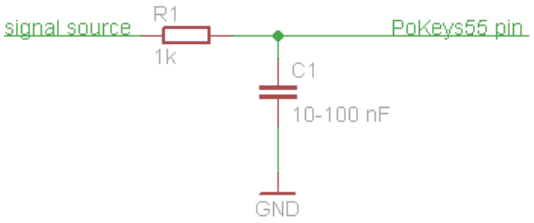
I have connected a switch to pin 54 and now PoKeys is connecting in recovery mode on every boot.

You must have connected normally-closed switch to pin 54 and therefore connected pin 4 to ground. At boot (connecting PoKeys device to USB) this means that PoKeys device is entering recovery mode and therefore cannot be used from the third-party software. Please use another pin for normally-closed switches. Also see chapter 9: Quick resetting the device configuration.

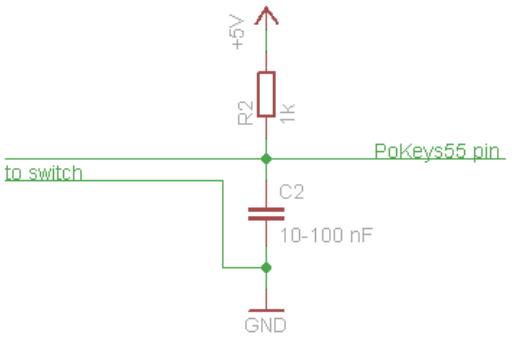
There is spontaneous triggering of some of the pins. What is wrong?

You might have connected long cable from the PoKeys board to the switch. If this cable crosses any power cables, it can trigger a false signal on PoKeys board input due to interference or coupling. Use twisted pair wires if possible. If this not help and cables cannot be routed elsewhere, use shielded cable.

If cables are routed inside electrically 'dirty' environment, use simple RC filter on those signals as illustrated below.



If connecting switches to PoKeys device with long cables, use the schematics below. When using matrix keyboard connected to PoKeys device with long cables, it is also advised to use the schematics below for digital input pins. This filter should not be used on digital output pins!



My problem is that when I start the program, everything works good for about 3 to 5 seconds, then the CPU "lock up" and the only way to recover is to unplug the PoKeys device.

The problem occurs because you are writing 'dirty' code. You create object every time you need to use it in a loop, but you forget do properly dispose it. Best way to use PoKeys DLL in an application that read or writes data in a loop, is to create a global object and initialize it once at the start of application, and use its functions to read or write in a loop. This way the communication is much faster.

If you are using Visual Basic development environment, add a reference to PoKeysDevice DLL and use object browser to find proper declaration. Via object browser you can also access the list of all supported functions, which will also be used by Intelli sense in editor.

Some outputs on PoExtBus boards are not responding.

If you encounter a problem with multiple PoExtBus boards not functioning normally, please check if the boards have the following 1 nF capacitor soldered between pins 2 (GND) and 3 (DATA) of the ExtBusOUT connector (see figure below).

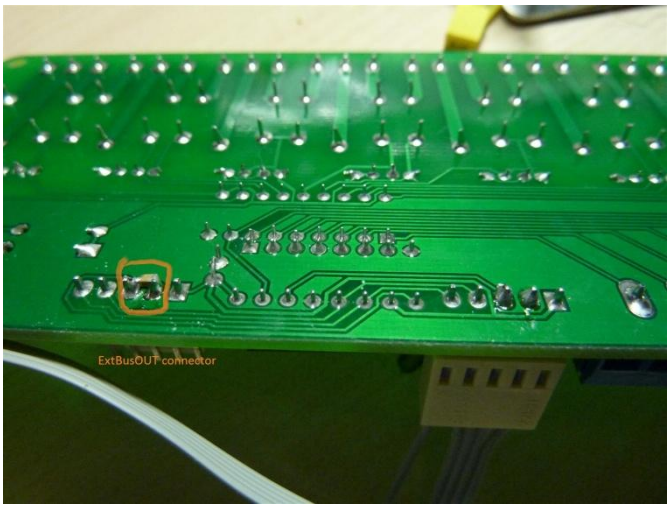


Figure 47: Capacitor on PoExtBus board

14. PoKeys library functions

The following list contains only the most commonly used library functions.

For the list and description of all library functions, open the PoKeys DLL help file found in Start Menu > PoKeys > > Development > PoKeys DLL help.

14.1. EnumerateDevices (for USB-based PoKeys devices)

Enumerate the PoKeys devices and return number of found PoKeys devices.

```
int EnumerateDevices()
```

Arguments:

none

Remarks:

This function must be called on every class initialization

14.2. ConnectToDevice (for USB-based PoKeys devices)

Connect to the device with the index specified.

```
bool ConnectToDevice(int deviceIndex)
```

Arguments:

deviceIndex

Index of the PoKeys device

Remarks:

Index is not UserID of the PoKeys device and therefore can change if more than one PoKeys device is used at a time. Function returns True if connection is established or False if there were errors.

14.3. DisconnectDevice

Terminate the connection with the device.

```
void DisconnectDevice()
```

Arguments:

none

Remarks:

This function should be called before class disposal or changing of the device.

14.4. StartEthernetDiscovery

Starts the automatic discovery of PoKeys56E devices.

```
void StartEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

14.5. StopEthernetDiscovery

Stops the automatic discovery of PoKeys56E devices.

```
void StopEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

14.6. GetNumberOfDetectedNetworkDevices

Returns the number of detected PoKeys56E devices.

```
int GetNumberOfDetectedNetworkDevices()
```

Arguments:

none

Remarks:

See example section for usage example

14.7. ConnectToNetworkDevice

Connects to PoKeys56E device with the specified IPv4 address in format xxx.xxx.xxx.xxx (e.g., 192.168.0.105, 192.168.050.002, etc.). If connection is established successfully, returns True.

```
bool ConnectToNetworkDevice(string address)
```

Arguments:

address

IPv4 address of the selected device

Remarks:

See example section for usage example

14.8. GetDeviceID

Retrieve device ID data, i.e. serial number and firmware version.

```
bool GetDeviceID(ref int serialNumber, ref int firmwareVersion, ref int pinNum)
```

Arguments:

serialNumber

Variable in which serial number will be saved to

firmwareVersion

Variable in which firmware version will be saved to

pinNum

Variable in which number of pins will be saved to

Remarks:

It is advised to use GetDeviceIDEx instead of this function. Returns False on error.

14.9. GetDeviceIDEx

Retrieve device ID data, i.e. serial number and firmware version.

```
bool GetDeviceIDEx(ref int serialNumber, ref int firmwareVersionMajor, ref int firmwareVersionMinor)
```

Arguments:

serialNumber
Variable in which serial number will be saved to

firmwareVersionMajor
Variable in which firmware major version will be saved to

firmwareVersionMinor
Variable in which firmware minor version will be saved to

Remarks:

Returns False on error.

14.10. GetBuildDate

Retrieve firmware build date.

```
bool GetBuildDate(ref string buildDate)
```

Arguments:

buildDate
Variable in which build date will be saved to

Remarks:

Returns False on error.

14.11. GetUserID

Retrieve user ID.

```
bool GetUserID(ref byte userID)
```

Arguments:

userID
Variable in which user ID will be saved to

Remarks:

Returns False on error.

14.12. SetUserID

Set user ID.

```
bool SetUserID(byte newUserID)
```

Arguments:

newUserID

New user ID

Remarks:

It is advised that each PoKeys device on a system should have its unique user ID. Returns False on error.

14.13. SetPinData

Set pin data – pin's function and options.

```
bool SetPinData(byte pinID, byte pinFunction, byte pullUpDownResistor, byte invertPin)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

Remarks:

It is advised to use function SetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

14.14. SetPinData

Set pin data – pin's function and options.

```
bool SetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

14.15. GetPinData

Get pin data – pin's function and options.

```
bool GetPinData(byte pinID, ref byte pinFunction, ref byte
pullUpDownResistor, ref byte invertPin, ref byte pinPossibleFunctions)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

pinPossibleFunctions

No function at the moment

Remarks:

It is advised to use function GetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

14.16. GetPinData

Get pin data – pin's function and options.

```
bool GetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

14.17. GetInput

Get digital input status.

```
bool GetInput(byte pinID, ref bool inputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputState

Variable in which input state will be saved to

Remarks:

Pin must be set as digital input for this command to function properly. Returns False on error.

14.18. SetOutput

Set digital output status.

```
bool SetOutput(byte pinID, bool outputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

outputState

New output state

Remarks:

Pin must be set as digital output before this operation will function properly. On device initialization, pins are NOT set as outputs even if they are configured so. Before using the pins as digital outputs SetPinData must be called to set the direction of the pin. This must be repeated on every startup. Returns False on error.

14.19. GetAnalogInput

Get analog input status.

```
bool GetAnalogInput(byte pinID, ref [int,byte] inputValue)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputValue

Variable in which analog value will be saved to. Use byte type for 8-bit analog values and int type for 10-bit (or 12-bit for PoKeys56 devices) analog values

Remarks:

The returned value is between 0 and 255 (8-bit resolution) , between 0 and 1023 (10-bit resolution on PoKeys55 devices) or between 0 and 4095 (12-bit resolution on PoKeys56 devices). 0 means 0 V on input, while 255, 1023, respectively 4095 means Vdd (approximately 3.3V) on input. Returns False on error. Use the command GetAnalogValueMax to get this value automatically.

GetAnalogValueMax

Gets the maximum value of the analog-to-digital conversion. Returns 1024 for PoKeys55 devices and 4096 for PoKeys56 devices.

```
int GetAnalogValueMax()
```

Arguments:

none

14.20. SetAnalogOutput

Get analog input status.

```
bool SetAnalogOutputValue (byte pinID, [int,byte] value)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

value

New 8-bit or 10-bit value for analog output.

Remarks:

The value specified must be between 0 and 255 (8-bit resolution) or 0 and 1023 (10-bit resolution). 0 means 0 V on output, while 255 respectively 1023 means Vdd (approximately 3.3V) on output. Returns False on error.

14.21. SaveConfiguration

Saves current device configuration to non-volatile flash memory.

```
bool SaveConfiguration()
```

Arguments:

none

Remarks:

This function takes some time to complete. Is meantime, communication with the device is not possible. Returns False on error.

14.22. GetMatrixConfiguration

Get complete matrix keyboard configuration.

```
bool GetMatrixKeyboardConfiguration(ref byte configuration, ref byte width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row. Row pins must be set as digital outputs.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column. Column pins must be set as digital inputs.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	...					

Remarks:

Row pins must be set as digital outputs and column pins as digital inputs respectively. Returns False on error.

14.23. SetMatrixConfiguration

Set complete matrix keyboard configuration.

```
bool SetMatrixKeyboardConfiguration(ref byte configuration, ref byte width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	...					

Remarks:

Returns False on error.

14.24. GetPWMOutputs

Get complete PWM outputs configuration

```
bool GetPWMOutputs(ref bool[] channels, ref uint period, ref uint[] duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56 devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

14.25. SetPWMOutputs

Set complete PWM outputs configuration

```
bool SetPWMOutputs(ref bool[] channels, ref uint period, ref uint[] duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56 devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

14.26. LCDSetSettings

Set LCD settings.

```
bool LCDSetSettings(byte option, byte rows, byte cols)
```

Arguments:

option

Set to 1 to enable LCD support.

rows

number of rows on LCD module used.

cols

number of columns on LCD module used.

Remarks:

Enabling LCD support does NOT disable pins. Special care should be taken when simultanously using LCD designated pins for LCD display and digital inputs or outputs.

14.27. LCDInit

Initializes the LCD module.

```
bool LCDInit()
```

Arguments:

No arguments needed

Remarks:

No remarks.

14.28. LCDClear

Clears LCD display.

```
bool LCDClear()
```

Arguments:

No arguments needed

Remarks:

No remarks.

14.29. LCDGotoXY

Move cursor to position x (column), y (row)

```
bool LCDGotoXY(byte x, byte y)
```

Arguments:

x

Column

y

Row

Remarks:

x and y are 1-based addresses. Home position therefore is at 1,1.

14.30. LCDPutc

Displays one character on LCD module.

```
bool LCDPutc(char character)
```

Arguments:

character

Character code of character to be displayed

Remarks:

Character is displayed from at current position of the cursor.

14.31. LCDPrint

Displays string on LCD

```
bool LCDPrint(string LCDText)
```

Arguments:

LCDText

String containing up to 20 characters and \0 termination.

Remarks:

Text is displayed from the current position of the cursor.

14.32. LCDSetEntryMode

Set option for 'Entry mode'.

```
bool LCDSetEntryMode(byte CursorMoveDirection, byte DisplayShift)
```

Arguments:

CursorMoveDirection

0 for left-to-right movement, 1 for right-to-left movement

DisplayShift

0 for display shift off, 1 for display shift on

Remarks:

No remarks.

14.33. LCDDisplayOnOffControl

Set option for 'On/off control'.

```
bool LCDDisplayOnOffControl(byte DisplayOnOff, byte CursorOnOff, byte
CursorBlinkingOnOff)
```

Arguments:

DisplayOnOff

1 for display on.

CursorOnOff

1 for cursor display on.

CursorBlinkingOnOff

1 for cursor blinking on

Remarks:

No remarks.

14.34. LCDDefineCustomCharacter

Defines one custom character.

```
bool LCDDefineCustomCharacter(byte CharacterCode, byte[] characterData)
```

Arguments:

Character code

Character code in range from 0 to 7

characterData

8-byte array containing custom character data as specified in LCD module documentation.

Remarks:

No remarks.

14.35. AuxilaryBusGetData

Reads PoExtBus enabled setting.

```
bool AuxilaryBusGetData(ref byte auxilaryBusEnabled)
```

Arguments:

auxilaryBusEnabled

PoExtBus enabled status (if 1, PoExtBus is enabled, otherwise disabled)

Remarks:

No remarks.

14.36. AuxilaryBusSetData

Sets-up PoExtBus settings and data.

```
bool AuxilaryBusSetData(byte auxilaryBusEnabled, byte[] dataBytes)
```

Arguments:

auxilaryBusEnabled

PoExtBus enabled status (if 1, PoExtBus is enabled, otherwise disabled)

dataBytes

Array of 10 byte values that are sent to the bus. First device on the PoExtBus always receives the 10th element of the array, the second device (if present) receives 9th element of the array, etc.

Remarks:

No remarks.

15. Interfacing with PoKeys library – C# example

Preinitialization

C#

1. Add a reference to a PoKeysDevice_DLL.dll, located in installation folder
2. Use the class `PoKeysDevice` from the PoKeysDevice_DLL namespace

Visual Basic 6.0

1. Add a reference to a PoKeysDevice_DLL.tlb, located in installation folder
2. Use the class `PoKeysDevice_DLL.PoKeysDevice`

Class initialization

```
PoKeysDevice_DLL.PoKeysDevice cPoKeys = new PoKeysDevice_DLL.PoKeysDevice();
```

Connecting to USB devices – PoKeys55/PoKeys56U

Enumerating the USB devices (this step must be taken even if we know exact device user ID!)

```
int iNumDevices = cPoKeys.EnumerateDevices();
```

The command returns the number of USB PoKeys devices detected on the system.

Getting device's serial number, user ID, firmware version and pin count:

```
int iSerialNumber = 0;
int iFirmwareVersion = 0;
int iPinNum = 0;
byte iUserID = 0;


for (int n = 0; n < iNumDevices; n++)
{
   cPoKeys.ConnectToDevice(n);
   cPoKeys.GetDeviceID(ref iSerialNumber, ref iFirmwareVersion, ref iPinNum);
   cPoKeys.GetUserID(ref iUserID);
   cPoKeys.DisconnectDevice();

   Console.WriteLine(n + ". device: Serial: " + iSerialNumber + "     Firmware: " +
iFirmwareVersion + "  User ID: " + iUserID);
}
```

Before any data can be read from or written to the device, the command ConnectToDevice must be executed. It's parameter is a device's index and not the userID! (therefore can be changed when multiple devices are connected at a time).

Connecting to Ethernet devices – PoKeys56E

Enumerating the ethernet devices, connecting and retrieving basic data

The example below enumerates PoKeys56E devices, connects to every one of them and displays its user ID and type in the listbox named lstEthDevices.

```
// Enumerate and list ethernet devices
MyDevice.StartEthernetDiscovery();
byte userID = 0;

lstEthDevices.Items.Clear();
lstEthDevices.Items.Add("Searching....");
lstEthDevices.Enabled = false;
lstEthDevices.Refresh();
Application.DoEvents();
Thread.Sleep(2000);

MyDevice.StopEthernetDiscovery();

if (MyDevice.GetNumberOfDetectedNetworkDevices() > 0)
{
   lstEthDevices.Enabled = true;
   lstEthDevices.Items.Clear();
   cmdEthConnect.Enabled = true;
}
else
{
   lstEthDevices.Items.Clear();
   lstEthDevices.Items.Add("No device detected");
   cmdEthConnect.Enabled = false;
}
for (int k = 0; k < MyDevice.GetNumberOfDetectedNetworkDevices(); k++)
{
     // Three possibilites to connect to network device...
    MyDevice.ConnectToNetworkDevice(MyDevice.EthDeviceIP[k]);

    // Also
    //MyDevice.ConnectToNetworkDevice(MyDevice.GetNetworkDeviceAddress(k));

    // or even (with pointers)
    //PoKeysDevice_DLL.sPoKeysNetworkDeviceData netData = new
PoKeysDevice_DLL.sPoKeysNetworkDeviceData();
    //IntPtr ptr =  Marshal.AllocHGlobal(200);
    //Marshal.StructureToPtr(netData, ptr, false);

    //MyDevice.COM_GetNetworkDeviceData(ptr, 0);

    //netData = (PoKeysDevice_DLL.sPoKeysNetworkDeviceData)Marshal.PtrToStructure(ptr,
typeof(PoKeysDevice_DLL.sPoKeysNetworkDeviceData));
    //MyDevice.ConnectToNetworkDevice(netData.IPAddress);

   MyDevice.GetUserID(ref userID);
   lstEthDevices.Items.Add(MyDevice.GetDeviceTypeName() + " (" + userID + ")");

   MyDevice.DisconnectDevice();
}
```

Remarks: the array `MyDevice.EthDeviceIP` holds the IP addresses of all detected devices in the process of automatic discovery.

PoKeys56E has a short connection timeout setting of 3 seconds (default, but can be altered in the Device>Network device settings... menu. After no data is received for this period, the connection with the host is terminated.

Reading pin configuration

```
byte iPinFunction = 0;

cPoKeys.GetPinData(0, ref iPinFunction);
```

In this example 0 (Pin 1) is used for a pin ID. Pin IDs are 0 based.

iPinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Reading pin key mapping (only on USB PoKeys devices – PoKeys55 and PoKeys56U)

Let us presume that pin 2 is defined as keyboard digital input with direct key mapping

```
byte iPinKey = 0;
byte iPinModifier = 0;
byte iMappingType = 0;

cPoKeys.GetPinKeyMapping(1, ref iMappingType, ref iPinKey, ref iPinModifier);
```

iPinKey is a key code as described in USB HID standard
iPinModifier is a modifier for a key (Ctrl, Alt, Shift, Win key) and can be used with these masks:

```
const byte CtrlMask = 1;
const byte ShiftMask = 2;
const byte AltMask = 4;
const byte WinMask = 8;
const byte AltGrMask = 64;
```

iMappingType has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	reserved	macro	direct	enable

bit 0 – Enable key mapping: to enable key mapping for a specific pin, this bit must be set to 1
bit 1 – enable direct key mapping: when this bit is set to 1, pin actions are directly reflected as a keyboard key
bit 2 – enable keyboard macro mapping: when this bit is set to 1, special macro sequence is sent on pin activation

There can be only one of the bits 1 or 2 set!

Reading input value

Let us presume that pin 3 is configured as digital input.

```
bool bInputVal = false;
cPoKeys.GetInput(2, ref bInputVal);
Console.WriteLine("Input 3 is " + (bInputVal?"On":"Off"));
```

Reading analog input value

Let us presume that pin 43 is configured as analog input.

```
// Read analog value
int value = 0;
MyDevice.GetAnalogInput(42, ref value);

MessageBox.Show("Analog value: " + (3.3 * (double)value /
     MyDevice.GetAnalogValueMax()).ToString("0.00") + " V");
```

Joystick axis mapping (only on USB PoKeys devices – PoKeys55 and PoKeys56U)

This is only possible on pins 43-47. If this is used on any other pin, the function will fail or be ignored.

```
cPoKeys.SetJoystickAxisMapping(42, iJoystickAxis);
```

iJoystickAxis can be used as follows:

```
0  None
1  Rx
2  Ry
3  X
4  Y
5  Throttle
```

Block read - digital

It is possible to poll 32 input pins with one request. All 55 pins can be read with two request joined in single command.

```
// Read pins 1 to 32
bool[] values_1_32 = new bool[32];

myDevice.BlockGetInput1(ref values_1_32);

// Read pins 33 to 55
bool[] values_33_55 = new bool[23];

myDevice.BlockGetInput2(ref values_33_55);

// Read all pins (1-55)
bool[] values = new bool[55];

myDevice.BlockGetInputAll55(ref values);
```

Block read - analog

It is possible to poll 4 8-bit or 3 10-bit analog inputs with one command.

```
// 8-bit mode
byte[] channels = { 42, 43, 0, 45 };
byte[] values = new byte[channels.Length];

myDevice.BlockGetAnalogInput8bit(ref channels, ref values);

byte value1 = values[0];
byte value2 = values[1];
byte value3 = values[3];

// 10-bit mode
byte[] channels = { 42, 43, 45};
int[] values = new int[channels.Length];
```

```
myDevice.BlockGetAnalogInput10bit(ref channels, ref values);

int value1 = values[0];
int value2 = values[1];
int value3 = values[2];
```

Block write - digital

It is possible to set 32 output pins with one request. All 55 pins can be set with two request joined in single command.

```
// Simple 8-bit binary counter
bool[] states = new bool[32];

for (int n = 0; n < 255; n++)
{
  for (int i = 0; i < 8; i++)
  {
    if ((n & (1 << i)) > 0) states [i] = false; else states [i] = true;
  }
  MyDevice.BlockSetOutput1(ref states); // Update pins 1 to 32
}
```

Reading encoder RAW values

RAW values from the encoder inputs can be read with following command.

```
byte iEncoderValue = 0;

cPoKeys.GetEncoderValue(1, ref iEncoderValue);
```

iEncoderValue is a value between 0 and 255.

Macro operations (only on USB PoKeys devices)

Create new macro

This command creates macro in first free position. It returns macro index.

```
byte iMacroID = 0;
byte iMacroLen= 10;

cPoKeys.MacroCreate(iMacroLen, ref iMacroID);
```

Modify macro length

This command modifes macro length.

```
byte iMacroID = 0;
byte iMacroNewLen = 50;

cPoKeys.MacroModifyLength(iMacroID, iMacroNewLen);
```

Delete macro

This command deletes specific macro.

```
byte iMacroID = 0;

cPoKeys.MacroDelete(iMacroID);
```

Save macro configuration to flash

This command saves the current macro configuration to flash.

```
cPoKeys.MacroSaveConfiguration();
```

Change macro name

This command changes the macro name. Name property supports up to 7 characters.

```
byte iMacroID = 0;
cPoKeys.MacroSetName(iMacroID, "Macro1");
```

Set macro key

This command sets one macro key at the position iIndex. This index must be between 0 and iMacroLen - 1.

```
byte iIndex = 5;
byte iKeyCode = 10;
byte iKeyModifier = 0;

cPoKeys.MacroSetKey(iMacroID, iIndex, iKeyCode, iKeyModifier);
```

Get free space for macros

Space for saving macros is limited. To find out how much free space exists, use the following command.

```
int iFreeSpace = 0;

cPoKeys.MacroGetFreeSpace(ref iFreeSpace);
```

Get the list of macros' states

If the macro has the length of 0 it is designated as inactive. This command retrieves the list of states for all the macros. If specific macro is active, bActiveMacros has the value True.

```
bool[] bActiveMacros = new bool[64];

cPoKeys.MacroGetActiveMacros(ref bActiveMacros);
```

Display current time on LCD

```
// Initialize library
PoKeysDevice_DLL.PoKeysDevice dev = new PoKeysDevice_DLL.PoKeysDevice();

// Enumerate devices and connect to first (we have only 1 connected)
dev.EnumerateDevices();
dev.ConnectToDevice(0);

// Set settings for 4x20 LCD
dev.LCDSetSettings(1, 4, 20);
// Initialize LCD
dev.LCDInit();
// Clear LCD
dev.LCDClear();
// Move cursor to home
dev.LCDGotoXY(1, 1);
// Print Hello, world!
dev.LCDPrint("Hello, world!");

// 1s delay
System.Threading.Thread.Sleep(1000);
```

```
while (true)
{
  Application.DoEvents();
  System.Threading.Thread.Sleep(100);

  // Move cursor to home
  dev.LCDGotoXY(1, 1);
  // Print current date and time
  dev.LCDPrint(DateTime.Now.ToString());
}
```

Configuring PWM outputs

Configures pin 17 as PWM output with 20 ms period and 1.5 ms (7.5%) duty cycle. If a model RC servo signal input is connected to this pin, servo motor horn should position itself in the middle position.

```
bool[] channel = new bool[6];
uint[] duty = new uint[6];

// PWM base clock is 12 MHz (or 25 MHz on PoKeys56 devices), so 1 ms takes 12000 (25000)
cycles
float ms = MyDevice.GetPWMFrequency() / 1000;
uint period = (uint)(ms * 20);   // 20 ms period

channel[5] = true; // Pin 17 = channel 5 (Pin 18 = channel 4, ...)
duty[5] = (uint)(ms * 1.5); // Set duty cycle to 1.5 ms (7.5 %)

MyDevice.SetPWMOutputs(ref channel, ref period, ref duty);
```

Configuring matrix keyboard

Configures a 4x4 matrix keyboard, connected to first eight pins of PoKeys device.

```
byte[] rows = new byte[16]; // This must always be the size of 16
byte[] cols = new byte[8];  // This must always be the size of 8

bool[] macros = new bool[128];      // This must always be the size of 128
byte[] keys = new byte[128];         // This must always be the size of 128
byte[] modifiers = new byte[128];    // This must always be the size of 128

for (int i=0; i<4; i++)
{
    cols[i] = (byte)i;
    rows[i] = (byte)(i+4);
}

byte configuration = 1;
byte numrows = 4;
byte numcols = 4;

// Set column pins as digital inputs
MyDevice.SetPinData(0, 1 << 1);
MyDevice.SetPinData(1, 1 << 1);
MyDevice.SetPinData(2, 1 << 1);
MyDevice.SetPinData(3, 1 << 1);

// Set row pins as digital outputs
MyDevice.SetPinData(4, 1 << 2);
MyDevice.SetPinData(5, 1 << 2);
MyDevice.SetPinData(6, 1 << 2);
MyDevice.SetPinData(7, 1 << 2);

MyDevice.SetMatrixKeyboardConfiguration(ref configuration, ref numcols, ref numrows, ref
rows, ref cols, ref macros, ref keys, ref modifiers);
```

Reading matrix keyboard status

Reads all matrix keyboard keys statuses.

```
bool[] KeyStates = new bool[128];

MyDevice.GetMatrixKeyboardKeyStatus(ref KeyStates);

string status = "";
for (int row = 0; row < 4; row++)
{
    for (int col = 0; col < 4; col++)
    {
        status += KeyStates[row * 8 + col] ? "1 " : "0 ";
    }
    status += "\n";
}

MessageBox.Show("Key statuses:\n" + status);
```

Write data to PoExtBus device

Sends the byte to the first PoExtBus device. As the devices can be daisy-chained, the first device connected to PoKeys device has the index 9. If any device is daisy-chained to this device, it would have the index 8, etc.

```
byte[] dataOut = new byte[10];

// Set some value
dataOut[9] = 0xAA;

MyDevice.AuxilaryBusSetData(1, dataOut);
```

Write data to matrix LED display

This example activates the matrix LED display 1 as 8x8 display, clears it and draws a + sign on it.

```
// Enable LED1
MyDevice.MatrixLEDSetSettings(true, 8, 8, false, 8, 8);

// Invert the whole matrix
MyDevice.MatrixLED1ClearAll(true);

// Draw a dark + sign
MyDevice.MatrixLED1SetPixel(2, 1, false);
MyDevice.MatrixLED1SetPixel(1, 1, false);
MyDevice.MatrixLED1SetPixel(0, 1, false);
MyDevice.MatrixLED1SetPixel(1, 0, false);
MyDevice.MatrixLED1SetPixel(1, 2, false);
```

Read tick counter

Every PoKeys device has a tick counter, a counter that internally counts milliseconds.

```
uint ticks = 0;

MyDevice.GetTickCounter(ref ticks);

MessageBox.Show("Tick counter: " + ticks);
```

Read temperature from the LM75 sensor, connected to I²C bus

```
byte stat = 0;
byte[] data = new byte[32];

// Write 0 to the device at the address 0x90 (LM75 command: set address)
data[0] = 0;
MyDevice.I2CStartWrite(0x90, 1, data);
MyDevice.I2CGetWriteStatus(ref stat);
```

```
if (stat != 1)
{
    MessageBox.Show("Error writing");
    return;
}

// Read temperature - read two bytes from the device at the address of 0x90
MyDevice.I2CStartRead(0x90, 2);
// This operation was quite fast, so no wait is needed...
MyDevice.I2CGetReadStatus(ref stat, ref data);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + data[0] + "°C");
}
else
{
    MessageBox.Show("Error reading");
}
```

Read temperature from DS18B20 sensor, connected to 1-wire bus

```
byte stat = 0;
byte[] data = new byte[16];

MyDevice.prot1WireSetStatus(true);

data[0] = 0xCC;
data[1] = 0x44;
MyDevice.prot1WireStartWriteAndRead(2, 0, data);

Thread.Sleep(1000);

data[0] = 0xCC;
data[1] = 0xBE;
MyDevice.prot1WireStartWriteAndRead(2, 9, data);

Thread.Sleep(10);

byte len = 0;
MyDevice.prot1WireGetReadStatus(ref stat, ref data, ref len);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + (((data[1] << 8) + data[0]) >> 4) + "°C");
}
else
{
    MessageBox.Show("Error reading");
}
```

16. Major changes from 1.x to 1.7:

To move the PoKeys55 device to a new level some major changes to interface were imminent.

Pi n function 1 was removed (this was directly key mapped pin function). Instead, key mapping functionality was added to any digital input pin. Key mapping type (none/direct/macro) can be set via changed Key mapping command as shown in above example.

16.1. Pin 13 not functioning appropriately

On PoKeys55 devices with serial numbers greater than 10133 and lower than 11500 there is a flawed connection for pin 13. Please do not use this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

16.2. Putting pin 4 low on startup disables PoKeys55 device from booting.

Avoid connecting switches that can be closed on startup to this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

17. Errata information

This section describes special limitations of the device.

Pins 5 and 6 cannot be separately set as outputs/inputs

Pins 5 and 6 must be both set to either input or output

Affected: PoKeys56E

18. Grant of license

The material contained in this release is licensed, not sold. PoLabs grants a license to the person who installs this software, subject to the conditions listed below.

1. Access

The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.

Usage

The software in this release is for use only with PoLabs products or with data collected using PoLabs products.

Copyright

PoLabs claims the copyright of, and retains the rights to, all material (software, documents etc) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.

Liability

PoLabs and its agents shall not be liable for any loss or damage, howsoever caused, related to the use of PoLabs equipment or software, unless excluded by statute.

Fitness for purpose

No two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.

Mission Critical applications

Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this license specifically excludes usage in 'mission critical' applications, for example life support systems.

Viruses

This software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.

Support

No software is ever error-free, but if you are unsatisfied with the performance of this software, please contact our technical support staff, who will try to fix the problem within a reasonable time.

Upgrades

We provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.

Trademarks

Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56U, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.

support: www.poscope.com